



سال اول / شماره سوم / مرداد ماه ۱۳۸۸

برنامه نویسی

اولین مجله الکترونیکی جامعه برنامه نویسان

آموزشی زبان برنامه نویسی F# (قسمت دوم)

آیا برنامه نویسی خوبی هستید؟

Win32 در مقابل MFC

نصب و راه اندازی یک Mail Server در لینوکس

VOIP

CSS Sprite

نگاهی اجمالی بر هوک های ویندوز





متفاوت بیاندیشید متفاوت تبلیغ کنید...



شرکت پیشرو ارتباط بارمدا
پ . پ .

تلفن: ۶۴ ۷۳ ۸۴ ۷۷ - ۰۲۱

مجری انحصاری جذب آگهی مجله الکترونیکی برنامه نویس

<http://adv.barnamenevis.org>





نشریه الکترونیکی جامعه برنامه نویسان

شماره سوم / سال اول / مرداد ۱۳۸۸

| | |
|--|----|
| سخن سردبیر | ۴ |
| بررسی میزان حافظه مورد استفاده در برنامه‌های .NET | ۷ |
| آشنایی با سایت‌های برنامه نویسی | ۱۴ |
| معرفی ابزار | ۱۸ |
| نگاهی نزدیک تر به IP Telephony | ۲۲ |
| بررسی تکنیک CSS Sprites و استفاده از آن در ساختن Image Map | ۴۱ |
| معرفی کتاب | ۴۶ |
| ISAPI Extension چیست؟ | ۴۸ |
| قابلیت‌های جدید ASP.NET4.0 | ۵۸ |
| تعریف ویژگی‌ها (Properties) در جاوا اسکریپت | ۵۹ |
| آموزش زبان برنامه‌نویسی F# | ۶۱ |
| آیا برنامه‌نویس خوبی هستید؟ | ۶۸ |
| Win32 در برابر MFC | ۷۰ |
| آشنایی با بزرگان | ۷۶ |
| روپاه دوست‌داشتنی با امکاناتی جدید | ۷۷ |
| نصب و راه‌اندازی یک Mail Server کامل با qmail | ۸۱ |
| نگاهی اجمالی به هوک‌های ویندوز | ۸۹ |

و با تشکر از تمام کسانی که ما را در این شماره یاری کردند.

◀ مقالات و خبرهای درج شده در مجله؛ صرفاً نظر نویسندگان آنهاست و نظر مجله برنامه‌نویس نیست.

◀ مجله برنامه‌نویس در خلاصه کردن؛ ویرایش فنی و ادبی؛ حذف و اضافه مقالات مخیر است.

◀ تمام حقوق مادی و معنوی این اثر متعلق به مجله برنامه‌نویس و مدیر مسئول آن است. استفاده و نقل قول از مطالب این مجله با ذکر کامل منبع؛ بلامانع است.

◀ برای همکاری با مجله و یا ارسال انتقادات و پیشنهادات می‌توانید با ایمیل magazine@barnamenevis.org مکاتبه نمایید.

صاحب امتیاز:

جامعه برنامه نویسان فارسی زبان
www.barnamenevis.org

مدیر مسئول:

مهدی کرامتی فر

سردبیر:

مهدی عسگری

مدیر اجرایی:

حمیدرضا متقیان

طراح و صفحه آرا:

فهیمة زنجانی‌نژاد

طرح روی جلد:

صالح باقری

همکاران این شماره:

مهدی موسوی - بهروز راد
سعید موسوی فرید - حمیدرضا متقیان
حسین جزایری - محمد شمس جاوی
مهدی عسگری - محمد جاهد منش
سالار خلیل زاده

سازمان آگهی‌ها:

شرکت پیشرو ارتباط پارمیدا
تلفن:

۰۲۱-۷۷۸۴۷۳۶۴

۰۲۱-۷۷۲۶۹۴۷۳

وب سایت:

<http://adv.barnamenevis.org>

سخن سردبیر

مهدی عسگری

سلام

فکر کنم در لیست سخت‌ترین کارهای دنیا، پس از کارگری معدن، نوشتن سخن سردبیر باشد! نوشتن مطلبی یک صفحه‌ای درباره‌ی یک اثر که حاصل زحمت چندین نفر است کار زیاد آسانی نیست.

خوشحالم که در این زمان خاص به جای کلی کار دیگر که می‌توانستید انجام دهید دارید مجله "برنامه نویسی" را می‌خوانید. با فیدبک‌هایی که از شما خوانندگان عزیز مجله دریافت کردیم، سعی کردیم تمام ایرادهایی را که در شماره قبل به ما گوشزد کرده بودید برطرف کرده و علاوه بر آن کیفیت و کمیت مقالات را نیز در حد توانمان بالاتر ببریم (همین‌طور به مرور زمان نسبت به انتخاب مقاله‌ها سخت‌گیرتر می‌شویم که امیدوارم این موضوع در عمل قابل مشاهده باشد). ایمیل‌ها و پیام‌های انتقاد و پیشنهاد و تشکری که از شما دریافت کردیم بیش از هرچیز دیگر باعث دلگرمی‌مان شد.

برای یکی از خوانندگانمان اتفاق جالبی افتاد که برای من تلفنی تعریف کرد: این دوست عزیز که دانشجوی رشته‌ی کامپیوتر است صبح یکی از روزهای خرداد شروع به خواندن مجله کرده و غرق در آن می‌شود؛ وقتی که سرش را از مانیتور بر می‌دارد متوجه می‌شود که نه تنها نهار نخورده، بلکه زمان امتحانش نیز گذشته است. این اتفاق هم من را خوشحال کرد و هم ناراحت؛ ناراحت از بابت امتحان ندادن این دوست و خوشحال از این که مجله‌ای که من نیز نقش کوچکی در خلق آن داشتم جوانی را مجذوب خود کرده و برای وی کلی مطلب جدید و جذاب داشته است (وقتی تعریف‌های این چنینی از خوانندگان می‌شنوم خستگی و بدی‌های کار را فراموش کرده و فقط به این فکر می‌کنم که چطور می‌توانیم در شماره بعد اثر بهتری به شما تقدیم کنیم).

مسئله‌ای که این اواخر فکر مرا (همچون خیلی‌های دیگر) مشغول به خود کرده، نقش زنان و میزان فعالیت آنان در رشته نرم افزار (به ویژه برنامه‌نویسی) است. در دانشگاه بیشتر هم‌کلاسی‌هایم خانم بودند (بعضی وقت‌ها بیش از ۹۰ درصد)، اما در محیط کار کم‌تر برنامه‌نویس خانم می‌بینم؛ کم‌تر وبلاگ فنی از طرف یک خانم خوانده‌ام؛ کم دیده‌ام نرم‌افزارهای اپن سورسی که مؤلفشان یک خانم باشد و کلاً علی‌رغم توانایی‌های مساوی

زنان با مردان (و بعضی مواقع برتری آنان) در انجام کارهای فنی مهندسی، حضورشان کم‌رنگ‌تر از آقایان است (در اغلب کنفرانس‌های برنامه‌نویسی که دیده‌ام تعداد خانم‌های حاضر تکریمی بوده است؛ حتی در این مجله تا به حال مقاله‌ای از یک خانم نداشته‌ایم). سؤالی که در ذهن من وجود دارد این است که آیا زنان علاقه‌ای به برنامه‌نویسی و نرم‌افزار ندارند (که آن وقت باید پرسید آن تعداد زیاد فارغ التحصیل زن رشته‌ی مهندسی نرم‌افزار یا علوم کامپیوتر (در هر کشوری، نه فقط ایران) به چه کاری مشغول می‌شوند؟) و یا این که این حضور کم رنگ ریشه در تفاوت‌های روانی/بیولوژیکی زن و مرد دارد و خانم‌ها کم‌تر میل دارند دیده شوند؟

جواب هر چه که باشد، باید به راهکارهایی اندیشید تا زنان بیشتر در میدان عمومی برنامه‌نویسی و کلاً آی تی حضور داشته باشند. فکر کنم با من موافق باشید که الان تقریباً فضای برنامه‌نویسی و مهندسی نرم‌افزار مردمحور است (مثال‌ها:

<http://www.osnews.com/story/21803>

<http://www.ultrasaurus.com/sarahblog/2009/04/gender-and-sex-at-gogaruco/>

و کلی نمونه دیگر که به راحتی می‌توان با جستجویی ساده پیدا کرد و نشان از نادیده گرفتن حضور زنان در آی تی دارد)، حال خیلی دوست دارم که نظر شما را در این باره بدانم. من نیز به سهم خود در این زمینه دو کار می‌توانم بکنم:

۱- برای شماره بعد مطالعه و تحقیق کرده و مطلبی (نسبتاً) جامع درباره نقش زنان در دنیای مهندسی نرم‌افزار و راهکارهای افزایش مشارکت آنان در این فضا منتشر کنم (که نظرات شما در این باره بسیار موثر خواهد بود)

۲- علاوه بر دعوت به همکاری از تمام خوانندگان و علاقه‌مندان برای نوشتن مقاله برای ما، دعوت ویژه‌ای از خانم‌های برنامه‌نویس می‌کنم برای نشر آثارشان در این مجله و قول می‌دهم که مقالاتشان را در اولویت قرار دهم. در پایان از تمام بچه‌های مجله که در خلق این شماره مؤثر بودند و نیز شما خواننده عزیز که به این اثر معنی می‌بخشید، ممنونم.

مشتاقانه آماده شنیدن نظرات شما (پیشنهاد، انتقاد یا درخواست همکاری) هستیم:

magazine@barnamenevis.org

مهدی عسگری - تیر ماه ۱۳۸۸

سایت مرجع متخصصین ایران

www.IrExpert.ir



با بیش از ۱۰۶ هزار عضو متخصص

جهت عضویت رایگان
کلیک کنید

 IrExpert.ir



نوشته: Vance Morrison و Subramanian Ramaswamy

بررسی میزان حافظه مورد استفاده در برنامه‌های NET^۱

mehdi_mousavi@hotmail.com



مترجم: مهدی موسوی

اولین موردی که استفاده از حافظه اهمیت پیدا می‌کند، برنامه‌های CPU محوری است که داده‌های فراوانی را تغییر می‌دهند. یک کامپیوتر نوعی می‌تواند هر دستورالعمل را کمتر از نیم نانو ثانیه اجرا کند. البته این سرعت به مدت زمان درآوردن عملوندها از حافظه محدود می‌شود. پردازشگرهای امروزی، سلسله مراتبی از Cache دارند تا هزینه سخت افزار را کاهش دهند. Cache مرحله اول یا همان L1، سریع‌ترین حافظه است، اما نسبتاً کوچک است. سپس در این سلسله مراتب، L2 Cache قرار دارد که حافظه اصلی (RAM) بدنالش می‌آید و در نهایت، دیسک سخت. شکل ۱، زمان دستیابی و حجم بخش‌های مختلف سلسله مراتب حافظه یک PC را نشان می‌دهد. هر گامی که در عمق این سلسله مراتب پیش برویم، زمان دستیابی (و حجم) بصورت توان‌هایی از ۱۰ (یا بیشتر) افزوده می‌شود (دیسک سخت ۱۰۰۰۰ برابر کندتر از RAM است)، در حالی که هزینه (هر بایت) کاسته می‌شود.

بهینه‌سازی کارایی، تنها در مورد یک مطلب است: ساخت برنامه‌هایی که سریعتر اجرا می‌شوند. اجرای دستورالعمل‌ها برای سخت‌افزارهای پیشرفته امروزی آسان است، در حالی که بازیابی دستورالعمل‌های عملوند، هزینه‌بر است. بنابراین، استفاده از حافظه می‌تواند تأثیر مستقیم بر چگونگی سرعت اجرای برنامه داشته و معیاری مهم برای بهینه‌سازی باشد. در این مقاله، در مورد مفاهیم اولیه بهینه‌سازی استفاده از حافظه در برنامه‌های NET. سخن خواهیم گفت. ابتدا، مواردی که دستیابی به حافظه، گلوگاهی در نظر گرفته می‌شود و برای بهینه‌سازی مناسب است، بیان خواهد شد. سپس، به تفکیک خواهیم دید که حافظه چگونه در یک برنامه NET، استفاده می‌شود. در نهایت، در مورد ابزارها و استراتژی‌های اندازه‌گیری میزان استفاده از حافظه و کم کردن این میزان در برنامه‌های NET. صحبت خواهیم کرد.

مواردی که استفاده از حافظه بر سرعت تأثیر می‌گذارد:

شکل ۱ - حجم و زمان دسترسی در انبارهای غیر محلی

| | L1 Cache | L2 Cache | Memory (RAM) | Disk |
|-------------|----------|----------|--------------|--------------|
| Size | 64K | 512K | 2000M | 100,000M |
| Access Time | .4 ns | 4 ns | 40 ns | 4,000,000 ns |

اگر مسیرهای داده داغ به حافظه بیشتری دسترسی پیدا کنند، در آن‌صورت عملوندها اغلب باید از حافظه‌های کندتر برداشته شوند. از آنجایی که حافظه کندتر بصورت توان‌هایی از ۱۰ کند می‌شود، چند بار از دست رفتن L2 Cache می‌تواند تأثیر بسزایی بر کارایی داشته باشد.

مورد دومی که استفاده از حافظه اهمیت پیدا می‌کند، زمان شروع اولیه برنامه (Cold Startup) است. همانطور که شکل ۱ نشان می‌دهد، دستیابی به دیسک سخت بسیار کندتر از دستیابی به حافظه اصلی است. سیستم عامل سعی می‌کند با Cache کردن داده‌ها در حافظه اصلی این مسأله را تا حدی پوشش دهد. به این دلیل است که برنامه هنگام اجرای بار دوم، سریعتر است، فازی که به آن Warm Startup گفته می‌شود (داده‌ها در حافظه سریع‌تر ذخیره می‌شوند). هنگام شروع اولیه، عمل Cache کردن هنوز رخ نداده است و داده باید از روی

^۱- <http://msdn.microsoft.com/en-us/magazine/dd882521.aspx>

این روش‌ها، اغلب نیاز به تغییر روش‌های نمایش داده و تغییر زیادی از کدهای پیاده‌سازی شده دارند. بنابراین بهتر است این تغییرات را در چرخه توسعه زودتر انجام دهید تا پیشاپیش هزینه فکر کردن به حافظه را پرداخته باشید.

Task Manager

اولین گام در کاهش مصرف حافظه برنامه آنست که متوجه شوید برنامه شما چقدر حافظه مصرف می‌کند. بدین منظور می‌توانید از نرم افزار Task Manager موجود در ویندوز بهره‌مند شوید. اکثر کاربران با Task Manager آشنا هستند. شما می‌توانید آن را با نوشتن taskmgr در پنجره (WinKey + R)، یا فشار همزمان کلیدهای CTRL + ALT + DEL، و انتخاب Start Task Manager اجرا کنید. در Processes Tab، شما اطلاعاتی در مورد کلیه پروسه‌های در حال اجرای سیستم خواهید یافت. اگر پنجره حاوی ستون‌های Memory-Working، PID، Set و Memory-Private Working Set نیست، گزینه View و سپس Select Columns را انتخاب کرده و آن‌ها را به صفحه نمایش بیفزایید.

حافظه مشترک و غیر مشترک

مجموعه کاری (Working Set)، حافظه‌ای فیزیکی است که توسط پروسه در حال استفاده است. با این وجود، سیستم عامل عملیات بهینه‌سازی را به‌منظور اطمینان از عدم یکسان بودن هزینه نقاط مختلف حافظه انجام می‌دهد. بخش اعظمی از حافظه مورد استفاده توسط پروسه، داده‌های فقط خواندنی را نگهداری می‌کند (بعنوان مثال، دستور العمل‌های اجرایی). با توجه به اینکه این داده‌ها فقط خواندنی هستند، می‌توان آن‌ها را در بین کلیه پروسه‌هایی که بدان احتیاج دارند به اشتراک گذاشت. از آنجایی که کلیه پروسه‌ها از حافظه به اشتراک گذاشته شده فقط خواندنی کد سیستم عامل استفاده زیادی می‌کنند، بخش قابل توجهی از Working Set هر پروسه، مشترک است. بنابراین جمع کل Working Set، به نشان دادن دست بالای هزینه واقعی حافظه استفاده شده توسط یک پروسه، گرایش دارد.

سیستم عامل همچنین حافظه غیر مشترک (خصوصی) هر پروسه را نیز ردیابی می‌کند. این مسأله شامل حافظه‌های خواندنی نوشتنی مورد استفاده پروسه است. اگر چه این مجموعه غیر مشترک (Private Working Set) هزینه واقعی حافظه مورد استفاده یک پروسه را دست پایین می‌گیرد (چگونگی این مطلب را با استفاده از ابزار VADump در ادامه خواهیم دید)، با این حال معیار بهتری برای بهینه‌سازی است؛ زیرا بر خلاف بهینه‌سازی حافظه به اشتراک گذاشته شده، هرگونه بهبودی در حافظه خصوصی، فشار کلی حافظه روی ماشین را کاهش می‌دهد.

در نهایت، هر دو حافظه کلی و خصوصی، بخش مهمی از حافظه مورد استفاده توسط یک پروسه را نادیده می‌گیرد: File System Cache. از آنجایی که دسترسی به دیسک سخت بسیار هزینه بر

سخت دیسک برداشته شود. تنها روش برای بهبود این مسأله آنست که داده کمتری از دیسک سخت بارگذاری شود. تنها، حافظه‌ای که از دیسک برداشته می‌شود (مانند دستور العمل‌های برنامه) در شروع اولیه تأثیر دارد، حافظه مقداره‌ی شده توسط خود برنامه که شامل کلیه داده‌ها روی Heap و Stack می‌شود، تأثیری بر Startup ندارد.

آخرین موردی که استفاده از حافظه اهمیت پیدا می‌کند، هنگام تعویض برنامه‌ها (Application Switching) می‌باشد. هنگامی که برنامه شما بطور معقول بزرگ است (بزرگتر از 50MB) و کاربر به برنامه‌های دیگر سوئیچ می‌کند، این برنامه‌ها حافظه فیزیکی برنامه شما را می‌ربایند. هنگامی که کاربر به برنامه شما باز می‌گردد، آن صفحات ربوده شده باید از دیسک بازخوانی شوند که باعث کند شدن شدید برنامه شما می‌شود. این مسأله شبیه موضوع Cold Startup است، با این تفاوت که Cold Startup تنها بر روی دستور العمل‌های برنامه تأثیر نمی‌گذارد، بلکه بر روی کل حافظه (که حافظه مقداره‌ی شده توسط برنامه شما نیز شامل آن می‌شود) تأثیرگذار است. از آنجایی که سرورها برنامه‌های بسیار زیاد غیر مرتبطی را بطور همزمان و مداوم اجرا می‌کنند، سرورها مدام در حال application switching هستند و این بدان معناست که حافظه همواره موضوع مهمی برای سرورها است.

چه کار می‌توان انجام داد؟

اگر می‌توانستیم کد را با اعجاز بگونه‌ای بازچینی کنیم که کلیه درخواست‌ها از Cache های سریع پاسخ می‌گرفتند، برنامه به طرز چشمگیری سریع می‌شد. در عمل، این مسأله فقط در شرایط غیر معمول امکان پذیر است زیرا این الگوریتم برنامه است که ترتیب دسترسی به حافظه را تعیین می‌کند. تکنیکی که بیشتر قابل انجام است، حداقل کردن حافظه مورد استفاده است. این مسأله باعث کم شدن بار در Cache های سریع شده و باعث سرعت بیشتر برنامه می‌شود. برای ساختار داده‌هایی که حاوی بخش‌هایی (داغ) هستند که اغلب مورد دسترسی قرار می‌گیرند و در Cache پردازنده جا نمی‌گیرند (عموماً چنین داده‌هایی بیش از چند مگابایت هستند)، کاهش ۳۰٪ ای اندازه حافظه داده‌های داغ، عموماً منجر به افزایش ۱۰٪ ای سرعت پردازشگر می‌شود.

حافظه به ۳ طریق می‌تواند کم شود. اول آنکه شما می‌توانید کد کمتری اجرا کنید (که به Cold Startup کمک می‌کند). این مسأله در مواردی صادق است که چیزی با عدم کاردانی محاسبه شده باشد. دوم، می‌توانید به داده کمتری دست بزنید. این نیز شبیه استراتژی اول است، اما در مورد ساختمان داده‌های درگیر با مسأله بکار می‌رود. در نهایت (و احتمالاً عمومی‌ترین حالت)، ساختمان داده می‌تواند بصورت‌های متفاوت نوشته شده، کوچکتر شده، یا داده‌هایی (کوچک) که اغلب مورد دسترسی قرار می‌گیرند، از داده‌هایی (بزرگ) که کمتر مورد دسترسی قرار می‌گیرند، بصورت فیزیکی جدا شوند.

شما بزرگ‌تر باشد، بهینه‌سازی استفاده از حافظه در برنامه شما، احتمالاً با ارزش‌تر است.

روش ساده و سریع برای مشاهده میزان استفاده حافظه و بررسی هدر رفتن حافظه، اجرای "آزمون بویایی" است. برنامه را برای مدتی اجرا کنید و Working Set های آن را تحت نظر بگیرید. اگر Working Set های برنامه بدون محدودیت افزایش پیدا کردند، این به معنای هدر رفتن حافظه یا دیگر مشکلات مرتبط با حافظه است.

VADump: نگاهی جزئی‌تر

نرم افزار Task Manager تنها خلاصه‌ای از میزان استفاده یک برنامه از حافظه در اختیار قرار می‌دهد. برای دریافت جزئیات بیشتر نیاز به ابزاری با نام VADump دارید. این ابزار را می‌توانید با نوشتن VADump -sop ProcessID در پای خط فرمان در شاخه ای که VADump را در آن نصب کرده‌اید، اجرا کنید. این برنامه به تفکیک جزئیات حافظه مورد استفاده در یک پروسه را همراه DLL هایش به شما نشان می‌دهد. بخشی از این اطلاعات را در شکل ۲ می‌بینید:

است، حتی هنگامی که داده‌های فایل مستقیماً به حافظه Map نشده‌اند، توسط سیستم عامل Cache می‌شوند. این استفاده از حافظه، فشار حافظه بر سیستم را افزایش می‌دهد و در هیچ یک از معیارهای Working Set دیده نشده است (مالک این بخش سیستم عامل است). کار زیادی در مورد دسترسی به فایل نمی‌توان انجام داد (اگر برنامه شما به فایلی نیاز دارد، نمی‌توان از آن اجتناب کرد). بنابراین می‌توان آن را هزینه ای غیر قابل بهینه‌سازی قلمداد کرد.

حجم برنامه

هر برنامه را می‌توان بسته به میزان حافظه مورد استفاده‌اش در یکی از گروه‌های کوچک، متوسط یا بزرگ تقسیم‌بندی کرد. یک برنامه کوچک حاوی Working Set ای با حجم 20MB یا کمتر و Private Working Set ای کمتر از 5MB است. برنامه متوسط حاوی Working Set ای با حجم تقریبی 50MB و Private Working Set ای در حدود 20MB است. برنامه بزرگ عموماً حاوی Working Set ای با حجم بیش از 100MB و Private Working Set ای با حجمی بیش از 50MB است. هر چه برنامه

شکل ۲ - خروجی VADump باز شده توسط Notepad

| XmlView.vadump - Notepad | | | | | | |
|---|---------|-----------|---------|------------|------------|-----------------|
| File Edit Format View Help | | | | | | |
| Microsoft windows [version 6.0.6001] | | | | | | |
| Category | Total | | Private | Shareable | shared | |
| | Pages | KBytes | KBytes | KBytes | KBytes | |
| Page Table Pages | 106 | 424 | 424 | 0 | 0 | |
| Other system | 7 | 28 | 28 | 0 | 0 | |
| Code/StaticData | 4210 | 16840 | 944 | 5676 | 10220 | |
| Heap | 945 | 3780 | 3780 | 0 | 0 | |
| Stack | 33 | 132 | 132 | 0 | 0 | |
| Teb | 9 | 36 | 36 | 0 | 0 | |
| Mapped Data | 763 | 3052 | 0 | 132 | 2920 | |
| Other Data | 1750 | 7000 | 6996 | 4 | 0 | |
| Total Modules | 4210 | 16840 | 944 | 5676 | 10220 | |
| Total Dynamic Data | 3500 | 14000 | 10944 | 136 | 2920 | |
| Total system | 113 | 452 | 452 | 0 | 0 | |
| Grand Total working set | 7823 | 31292 | 12340 | 5812 | 13140 | |
| Module working set Contributions in pages | | | | | | |
| Total | Private | Shareable | Shared | Module | Preferred | |
| 17 | 2 | 15 | 0 | 0x00260000 | 0x00260000 | C:\dd\clr_1\src |
| 116 | 7 | 0 | 109 | 0x76E10000 | 0x76E10000 | C:\windows\sys |
| 30 | 3 | 4 | 23 | 0x71210000 | 0x71210000 | C:\windows\sys |
| 71 | 4 | 0 | 67 | 0x754C0000 | 0x754C0000 | C:\windows\sys |

به اشتراک گذاشته شده و قابل اشتراک تقسیم شده است. این چشم‌انداز به شما بدون شبهه نشان می‌دهد که آیا از بارگذاری DLL می‌توان منصرف شد و چند بایت از Private Working Set را می‌توان از Working Set برنامه زدود.

هنگامی که چنین DLL‌هایی شناسایی شدند (به‌عنوان نمونه، DLL‌ای که بدون استفاده در مسیر اجرایی خاصی از برنامه بارگذاری می‌شود)، قدم بعدی آنست که تشخیص دهیم که چرا چنین DLL بارگذاری می‌شود و بدنبال از بین بردن چنین بار ناخواسته‌ای باشیم. گام‌هایی که برای شناسایی DLL‌های مشکوک باید برداشت در وبلاگ CLR and Framework Perf ذکر شده است.

داده‌های Heap ای که در خروجی VADump نمایش داده شده اند، برای Heap‌های غیر مدیریت شده است (حافظه‌ای که توسط NET GC مدیریت نمی‌شود). مهم است که این مقدار را در سطح پایینی نگاه داریم تا GC بتواند بیشتر حافظه شما را با پاک کردن آن هنگام لزوم، مدیریت کند.

بخش Other Data (داده‌های دیگر) فراخوانی تابع بنیادی تخصیص حافظه سیستم عامل (VirtualAlloc) را نشان می‌دهد که VADump نمی‌تواند آن را بگونه‌ای دیگر طبقه‌بندی کند. برای برنامه‌های NET، عموماً مهمترین جزء Other Data بخش GC Heap است که کلیه اشیاء تعریف شده کاربر را نگهداری می‌کند.

NET Garbage Collector (زباله روب NET)

محیط اجرایی NET، از مدیریت خودکار حافظه پشتیبانی می‌کند. این بخش کلیه حافظه‌های تخصیص داده شده توسط برنامه مدیریت شده را ردیابی کرده و در بازه‌هایی GC را برای یافتن حافظه‌هایی که دیگر در حال استفاده نیستند، فراخوانی می‌کند. این حافظه‌ها سپس برای تخصیص‌های جدید مورد استفاده قرار می‌گیرند. بهینه‌سازی مهمی که GC انجام می‌دهد آنست که GC، کل حافظه را هر بار از ابتدا جستجو نمی‌کند، بلکه Heap را به سه نسل تقسیم می‌کند (0، 1 و 2).

نسل صفر کوچک‌ترین این نسل‌ها است و عموماً یک دهم میلی‌ثانیه زمان می‌برد تا تکمیل شود، اما تنها بررسی می‌کند که کدام تخصیص‌ها را که پس از آخرین GC رخ داده است (و البته در حال استفاده نیست)، باید پاک کند. در حالت ایده آل، حجم یک نسل کوچکتر از حجم L2 Cache است. نسل ۱، با تخصیص‌هایی سر و کار دارد که از یک GC، جان سالم به در برده‌اند. این نسل، مدت زمان بیشتری در مقایسه با GC نسل صفر به خود اختصاص می‌دهد؛ زمانی در حدود ۱ میلی‌ثانیه. در حالت ایده آل، باید بازای هر یک GC از نسل یک، ده GC از نسل صفر وجود داشته باشد. GC‌های نسل ۲ با کلیه اشیاء سر و کار دارند. بنابراین، زمان گرفته شده توسط این نسل ممکن است قابل توجه باشد. بعنوان مثال، ممکن است برای heap ای با ظرفیت 160، 20MB میلی‌ثانیه زمان صرف شود که زمان قابل توجهی است. زمان به‌صورت خطی با

برای خواندن این اطلاعات، از Grand Total Working Set آغاز کنید. این عدد با عدد نمایش داده شده در Task Manager باید یکسان باشد. این مقدار سپس به ۸ دسته شکسته شده است. دسته‌های قابل توجه عبارتند از:

- Code/Static Data که بیانگر DLL‌های بارگذاری شده پروسه است.
- Heap که بیانگر میزان حافظه Native (غیر GC) مصرف شده است.
- Other Data که نشان‌دهنده حافظه تخصیص داده شده توسط تابع VirtualAlloc سیستم عامل است. این مسأله برای کدهای Managed مهم است زیرا این بخش شامل کل GC Heap می‌باشد.

حافظه مورد استفاده DLL‌ها بازهم توسط VADump به بخش‌های کوچکتری پس از جدول اختصاری، تقسیم می‌شود. بازای هر DLL، هر بخش تعداد Page‌های مورد استفاده DLL را نشان می‌دهد (هر Page همواره 4K است). بنابراین می‌توان هزینه حافظه مصرفی کل کدی که بارگذاری شده است را تعیین کرد.

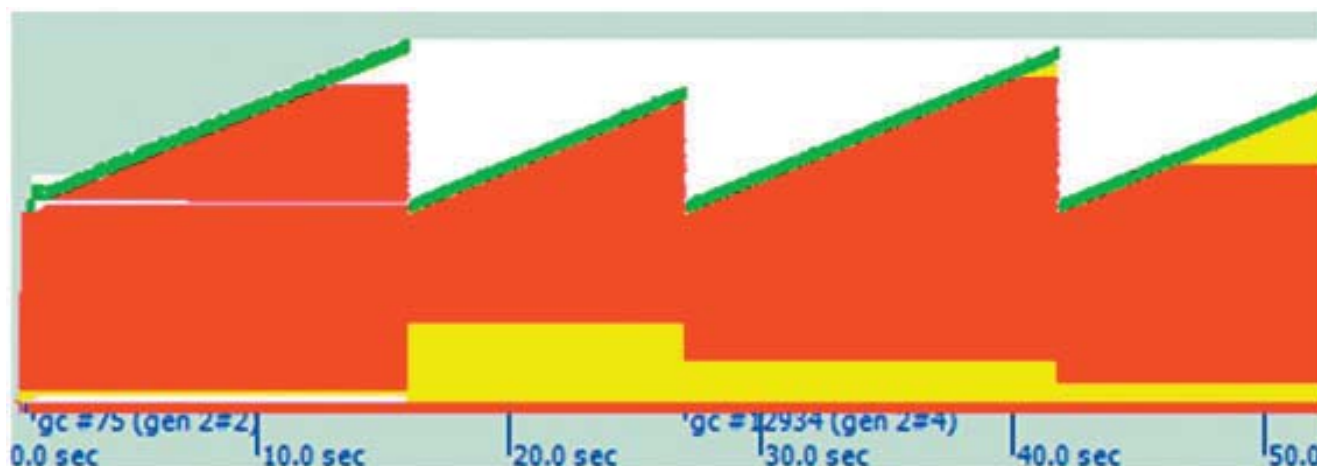
در شکل ۲، ردیفی تحت عنوان Grand Total Working Set وجود دارد که جمع کل Working Set را تحت مقیاس KB و تعداد Page‌ها را در ستون اول، نمایش می‌دهد. ستون‌های دوم، سوم و چهارم، Private Kbytes، Shareable Kbytes و SharedKbytes به ستون Total Working Set افزوده می‌شوند. این ستون دوم است که تحت عنوان Private Working Set در Task Manager نمایش داده می‌شود؛ در حالی که ستون اول، تحت عنوان Total Working Set در Task Manager نشان داده می‌شود. بنابراین VADump به شما امکان می‌دهد تا Private Working Set و Total Working Set را به‌همراه Working Set‌های قابل اشتراک گذاری و به اشتراک گذاشته شده، از یکدیگر تمیز دهید. این تصویر کاملتری از چیزی است که در Task Manager در دسترس است.

هنگامی که برنامه‌های NET بزرگ هستند، این بزرگی به دلیل اجرای زیاد کد یا استفاده زیاد از داده‌ها است. در چنین شرایطی، تعداد DLL‌های زیادی که بارگذاری شده‌اند را خواهید دید و بخش Code/Static Data تمایل به گرفتن کل Working Set را دارد. برای برنامه‌های Managed، این داده‌ها در GC Heap هستند و بنابراین تحت عنوان Other Data نشان داده می‌شوند، در حالی که بخش اعظمی از Working Set را گرفته‌اند.

در قسمت پایینی شکل ۲، Module Working Set‌هایی را خواهید دید که بر حسب واحد Page، لیست شده‌اند. این بخش به شما می‌گوید که کدام ماژول‌ها تحت قالب Working Set برنامه شرکت کرده‌اند و چه میزان از Working Set هر ماژول مصرف شده است. بنابراین می‌توانید به سرعت متوجه این مسأله شوید که Working Set یک DLL مشخص، به بخش‌های خصوصی،

افزایش حجم heap افزوده می‌شود (۸ ثانیه بازای هر مگابایت، اگر سختگیرانه تخمین زده باشیم). هزینه واقعی به مقدار حافظه باقی مانده، تعداد اشاره‌گرهای GC در حافظه باقیمانده و این که حافظه به چه میزان از هم گسسته است، بستگی دارد. در حالت ایده آل، بازای هر GC از نسل ۲، باید ۱۰ GC از نسل ۱ داشته باشیم.

در کل، همان‌طور که در شکل ۳ نشان داده شده است، Heap در NET GC، نمودار دندان اره‌ای است که سطح آن نمایانگر مجموعه‌های نسل ۲ است. نسبت heap به سطح در یک نسل دومی نوعی، چیزی در حدود ۱,۶ است، در حالی که این نرخ جدا از حجم heap (بدون گسستگی) می‌باشد. در صورت وجود گسستگی، این عدد ممکن است بطور قابل توجهی تغییر کند.

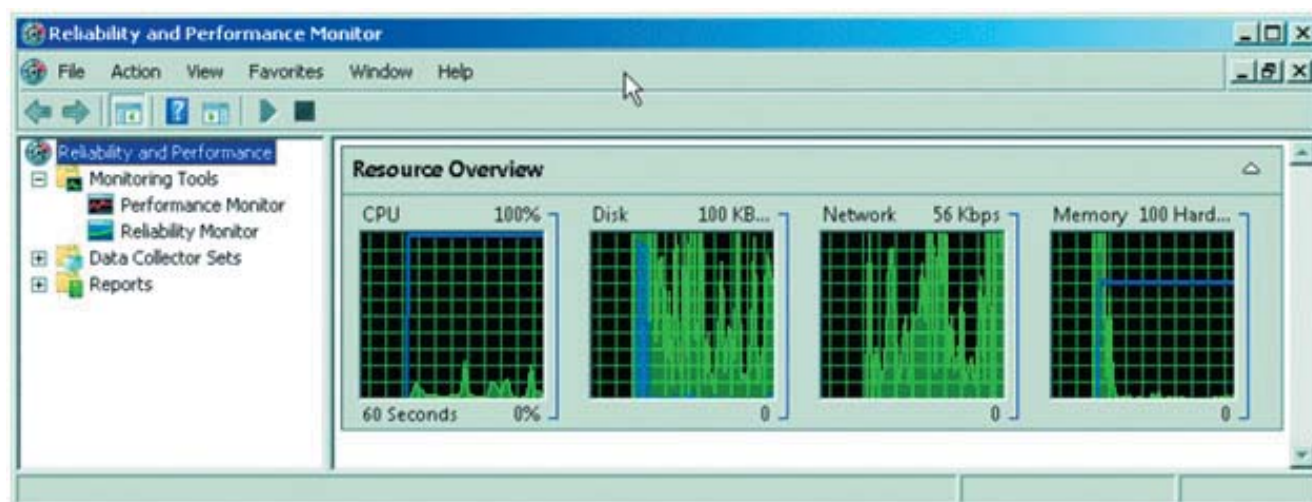


شکل ۳ - نمایش دندان اره‌ای GC Heap

Perfmon

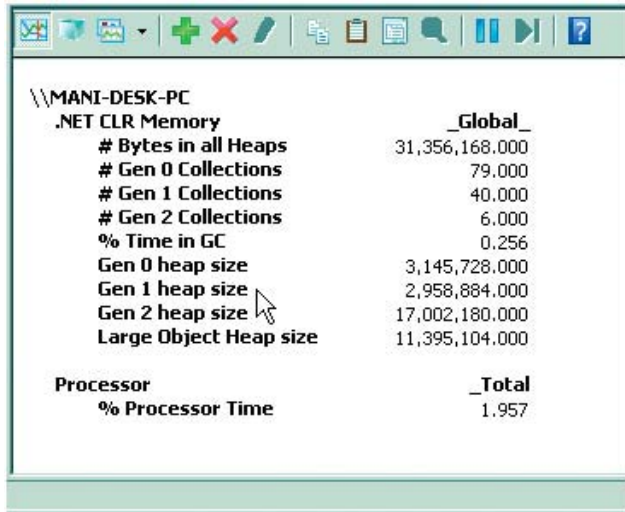
نرم افزار VADump اطلاعات سطح اولی از مصرف حافظه در یک پروسه را نشان می‌دهد. با این حال، بدقت به ما نمی‌گوید که چه میزان حافظه GC استفاده می‌کنیم (بخش Other Data می‌تواند شامل حافظه‌های دیگری غیر از GC Heap باشد). همچنین، به ما نمی‌گوید که آیا نرخ سالمی از نسل‌های GC داریم، یا خیر. به این دلیل، باید از نرم‌افزار Windows PerfMon استفاده کنیم. می‌توانید این برنامه را با نوشتن PerfMon در خط فرمان پنجره Run، اجرا کنید که باعث نمایش پنجره شکل ۴ می‌شود.

این نرم افزار قادر است تا داده‌های اجرایی با ارزشی جمع آوری کند، اما در این مقاله، تنها بر روی تحت نظر گرفتن GC Heap تمرکز می‌کنیم.



شکل ۴ - صفحه نخست PerfMon

در نهایت داده‌ها بصورت پیش فرض بصورت گرافیکی نشان داده می‌شوند، اما مطلوب‌تر است که آن‌ها را بصورت عددی نشان دهیم. این عمل را می‌توان با Click کردن روی نوار ابزار report-type (شکل ۶) انجام داد.



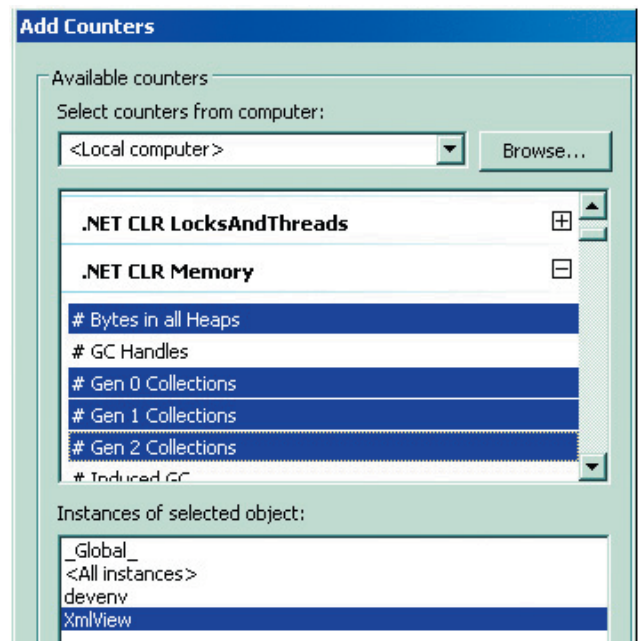
شکل ۶ - نمایش عددی PerfMon

با یک آزمون، متوجه شدیم که 7.3MB از کل 8.6MB حافظه Private Working Set، توسط GC Heap گرفته شده است و ۱۱ درصد از زمان در GC صرف شده است. مدت زمان مطلوب در GC کمتر از ۱۰ درصد کل زمان برنامه است، بنابراین این برنامه خاص روی مرز قرار گرفته است. سرانجام، این برنامه تعداد نسل‌های صفر، یک و دو را به ما گزارش می‌دهد. در حالت ایده آل، ما خواستار آن هستیم که تعداد نسل صفری‌ها، حداقل ۱۰ برابر نسل اولی‌ها، و نسل اولی‌ها ۱۰ برابر نسل دومی‌ها باشد.

نتیجه‌گیری

مسائل مرتبط با حافظه، به طرز بدنامی برای خطایابی دشوار است. اگر برنامه شما به میزانی بزرگ است که باید مراقب حافظه باشید، کلید موفقیت، محدود کردن استفاده از حافظه در همان فازهای ابتدایی توسعه است. درک این مطلب که حافظه چگونه به بخش‌های کوچکتر تقسیم می‌شود، اولین گام در این رویه است، درحالی‌که در نظر گرفتن میزان حافظه مورد استفاده گام بعدی می‌باشد. این مطالب را با سوالاتی از قبیل کدام DLL‌ها بیشترین حافظه را مصرف می‌کنند، چرا GC نسل ۲ اغلب اجرا می‌شود، آیا بارگذاری یک DLL مورد نیاز بوده است و ... به فرصت تبدیل کنید. سپس، مصرف حافظه را بطور مناسبی بهبود ببخشید. اگر تنها یک درس گرفته باشید، این نکته باید باشد که هر چه زودتر در چرخه توسعه به مسائل مربوط به حافظه توجه کنید، نتیجه‌اش را بعداً خواهید دید، بنابراین واقعاً به صرفه است که هر چه زودتر به حافظه فکر کنید.

پس از آن‌که پنجره PerfMon بالا آمد، باید آن‌را پیکربندی کنیم تا اطلاعات مربوط به GC را نمایش دهد. ما این کار را با Click کردن روی گزینه Performance Monitor در درخت کنترلی سمت چپ انجام می‌دهیم. این مسأله باعث تغییر بخش سمت راستی پنجره به‌منظور نمایش داده‌های شمارشی Performance می‌گردد. اکنون علامت + را برای افزودن شمارنده‌های جدید، فشار دهید. سپس، همان‌طور که در شکل ۵ نشان داده شده است، شمارنده‌ها و پروسه‌هایی که می‌خواهید تحت نظر بگیرید را انتخاب کنید.



شکل ۵ - انتخاب شمارنده‌ها برای زیر نظر گرفتن در PerfMon

هنگامی که چند شمارنده انتخاب کردید، متوجه اسامی همه برنامه‌هایی که از runtime مزبور استفاده می‌کنند، خواهید شد. شما می‌توانید یک، دو یا هر چند برنامه که می‌خواهید تحت نظر بگیرید را انتخاب کنید. علاوه بر این، گزینه‌ای تحت عنوان All وجود دارد که کلیه داده‌ها را بازای کلیه نمونه‌های نشان داده شده، تحت نظر می‌گیرد اما داده‌ها به تفکیک نشان داده می‌شوند. مضافاً، نمونه‌ای تحت عنوان _Global_ وجود دارد که داده‌های دیگر نمونه‌ها را با یکدیگر جمع می‌کند.

اگر برنامه‌ای پس از آغاز PerfMon برای تحت نظر گرفتن برنامه‌های دیگر اجرا شده بود، می‌توان با Click کردن روی علامت + و اضافه نمودن شمارنده‌های جدید برای برنامه‌های جدید، برنامه‌های جدیدی را نیز تحت نظر گرفت. (تنها، افزودن نمونه جدید مورد نیاز است، نمایش نمونه‌های دیگر در PerfMon متوقف نخواهد شد).

منابع بهره‌وری:

VA Dump:

<http://go.microsoft.com/fwlink/?LinkId=149683>

CLR Perf Team Blog (Instructions on investigating suspicious DLL loads):

<http://blogs.msdn.com/clrperfblog>

VS Profiler Team Blog:

<http://blogs.msdn.com/profiler>

Improving .NET Application Performance and Scalability:

<http://msdn.microsoft.com/library/ms998530>

Windows performance blog: Investigations using Xperf:

<http://blogs.msdn.com/pigscanfly/>

Vance Morrison's Blog:

<http://blogs.msdn.com/vancem>

Rico Mariani's Blog:

<http://blogs.msdn.com/ricom>

Lutz Roeder .NET Reflector for inspecting code:

<http://blog.lutzroeder.com>

CLR Inside Out - Investigating Memory Issues:

<http://msdn.microsoft.com/magazine/cc163528>

درباره نویسندگان:

Subramanian Ramaswamy، مدیر برنامه CLR Performance در مایکروسافت است. او دکترای برق و مهندسی کامپیوتر از موسسه فن آوری Georgia دارد.

Vance Morrison، دستیار طراح و مدیر گروه CLR Performance در مایکروسافت است. او طراحی زبان میانی .NET را به پیش برد و از ابتدای تشکیل .NET، در این زمینه فعالیت می‌کند.

www.DevTools.ir



مجموعه ای کامل از کامپوننتها، ابزارها، کتابها و مجموعه های آموزشی برای مهندسين نرم افزار و برنامه نويسان

مجموعه کامپوننت های Telerik

مجموعه کامپوننت های Dot NET

مجموعه ابزارهای Dot NET و مهندسی نرم افزار

مجموعه های آموزشی Multi Media

کتابهای مهندسی نرم افزار

Development Tools and Resources



مجموعه ای کامل از کامپوننتها، ابزارها، کتابها و مجموعه های آموزشی برای مهندسين نرم افزار و برنامه نويسان

www.DevTools.ir

www.stackoverflow.com

آشنایی با سایت های برنامه نویسی

نویسنده: مهدی عسگری

توانایی ویرایش پست‌های دیگران را دارند. در کنار این سایت دو سایت دیگر نیز به تازگی توسط این افراد درست شده به نام های superuser.com و serverfault.com که از اسمشان پیداست: اولی مربوط به سوال‌های کاربران حرفه‌ای است (نه برنامه‌نویسی) و سایت دوم مربوط به مدیران سیستم و سرورها.

در این سایت می‌توان به سوال‌ها امتیاز مثبت یا منفی داده و آن‌ها را پایین یا بالا برد؛ امتیاز هر سوال در کنار آن نوشته می‌شود. کسی که سوال را می‌پرسد می‌تواند انتخاب کند که کدام جواب، جوابی است که مد نظر وی بوده؛ برای همین ممکن است جوابی با امتیاز ۱۰+ پایین‌تر از جوابی با امتیاز ۵+ باشد.

سوال‌هایی که به عنوان جواب اصلی انتخاب می‌شوند با یک تیک سبز مشخص می‌شوند. افراد بر حسب سوال‌هایی که می‌پرسند، جواب‌هایی که می‌دهند، امتیازی که از دیگران می‌گیرند و ... ، بهشان reputation تعلق می‌گیرد که همانا امتیاز یا نشان‌دهنده سطح کاربر در سایت است (کاربران برتر سایت غالباً امتیازهای ۵ رقمی دارند)

در حال حاضر سایت حدود ۷۲۰۰۰ کاربر دارد.

در این شماره به معرفی سایت فوق و معماری و مشخصات سخت‌افزاری و نرم‌افزاری آن می‌پردازیم:

این سایت یک فروم برای پرسش و پاسخ درباره برنامه‌نویسی می‌باشد (هر زبانی، هر پلتفرمی)

این سایت در سال ۲۰۰۸ توسط Joel Spolsky و Jeff Atwood (که هر دو را حتماً می‌شناسید؛ قبلاً هم گفتم که وبلاگ این دو نفر جزو ۵ وبلاگ پربیننده برنامه‌نویسی جهان است که به ترتیب در آدرس‌های زیر واقع شده‌اند:

<http://joelonsoftware.com> و www.codinghorror.com

و در جواب به سایت‌های پولی‌ای مثل Experts Exchange به وجود آمد.

یکی از ویژگی‌های خوب این سایت این است که افراد خبره و مشهور نیز در آن فعالیت می‌کنند (هم به عنوان مدیر و هم کاربر معمولی) که باعث شده در مدتی کوتاه (کم‌تر از یک سال: از آگوست ۲۰۰۸ رسماً شروع به کار کرد) تبدیل به یکی از مهم‌ترین فروم‌های برنامه‌نویسی انگلیسی زبان شود. (از این افراد می‌توان به Walter Bright نویسنده زبان D ، Eric Lippert ، Jon Skeet نویسنده کتاب C# In Depth (اگر فکر می‌کنید برنامه‌نویس حرفه‌ای سی‌شارپ هستید

نگاهی به این کتاب بیندازید) و حتی افراد بسیار بزرگی مانند Alan Kay اشاره کرد). نکته مثبت دیگر این است که از هر چیزی که راجع به برنامه‌نویسی باشد در این سایت می‌توان سوال پرسید (محدودیتی ندارد، فقط کافی است به نوعی مربوط به برنامه نویسی باشد)

در حال حاضر علاوه بر Joel و Jeff ، سایت مدیران بسیاری دارد که بعضی از آن‌ها حتی پسورد سرورها را نیز در اختیار دارند؛ این مدیران از میان کاربران برتر سایت انتخاب شده‌اند و

The screenshot shows the Stack Overflow homepage. At the top, there's a navigation bar with links for 'Questions', 'Tags', 'Users', 'Badges', and 'Unanswered'. Below this, a 'Recent Questions' section lists several questions with their respective vote counts, answer counts, and view counts. For example, the first question is 'Setting the last-modified-time of a directory opened for ReadDirectoryChangesW' with 0 votes, 0 answers, and 9 views. To the right of the questions, there's a sidebar with 'Interesting Tags' and 'Ignored Tags' sections, each with an 'Add' button. At the bottom of the sidebar, there's a 'Your Ad Here.' section with a placeholder for an advertisement.

نمایی از صفحه اصلی سایت را در این عکس قرار داده‌ام (برای این که تب‌های دیگر فایرفاکس را نبینید، عکس را کات کرده‌ام!) سایت ۵ بخش اصلی دارد (بالا): Questions ، Tags ، Users ، Badges و Unanswered که قسمت آخر سوال‌های بدون جواب است. همان طور که در شکل می‌بینید بعضی از سوال‌ها در بخش featured قرار دارند: در این بخش سوال‌هایی وجود دارد که سوال پرسنده برای جواب دهنده جایزه گذاشته است، به این صورت که به کسی که جواب مد نظر وی را بدهد چند امتیاز (قابل تنظیم از طرف سوال پرسنده) از وی به جواب دهنده منتقل شود. (البته برای جلوگیری از زرنگ بازی بعضی‌ها، در صورتی که تا یک هفته پس از دریافت جواب‌ها، یکی را به عنوان جواب برتر انتخاب نکند، امتیاز به صورت خودکار به حساب جواب دهنده‌ای که جوابش بیشترین امتیاز را داشته، می‌رود). این فروم تالار ندارد و برای جستجو در بین سوال‌ها باید از tag ها و نیز قسمت Search آن بهره ببرید (امکان درست کردن این همه تالار وجود ندارد). در صورتی که می‌خواهید در مورد یکی از جواب‌ها اظهار نظر کنید می‌توانید برای آن جواب (پست) کامنت بگذارید و حتی به کامنت‌ها نیز می‌توان vote داد.

وبلاگ مدیران سایت در <http://blog.stackoverflow.com> واقع است. همچنین هر از چندگاهی (بین ۷ تا ۱۵ روز یک بار) Jeff و Joel یک podcast منتشر می‌کنند که از وبلاگشان قابل دسترسی است و در آن به موضوعات مختلف برنامه‌نویسی می‌پردازند و اغلب مهمان‌هایی نیز دارند. (<http://blog.stackoverflow.com/category/podcasts>). تا به این لحظه ۶۱ پادکست ضبط شده است) این هم عکسی از یکی از صفحات سایت:

امتیاز (vote) مرتب شده و نشان داده می‌شوند، اما می‌توان آن را بر حسب تاریخ جواب‌های داده شده به صورت صعودی یا نزولی مرتب کرد (oldest و newest). در ضمن می‌بینید که سوال توسط shog9 ویرایش شده و توسط hakim پرسیده شده و ویرایش پست‌های دیگران است و 19.7K یا 19700 امتیاز دارد). همین‌طور مشاهده می‌شود که کدها قابلیت syntax highlighting دارند. Joel تخمین می‌زند که حدود یک سوم برنامه‌نویسان جهان حداقل یک بار از این سایت استفاده کرده‌اند.

همان‌طور که می‌بینید (سمت راست) سوال پرسیده شده با tag های C ، Linux ، SMTP و DNS برچسب خورده و جوابی که به این سوال داده شده و به عنوان جواب درست انتخاب شده ۳ رای مثبت گرفته (شاید هم مثلاً ۴ رای مثبت و یک رای منفی گرفته) که با تیک سبز مشخص است. همچنین می‌بینید که کاربر Dave Rigby و Alnitak برای پست مربوط به سوال کامنت گذاشته‌اند. آن علامت ستاره‌ای که می‌بینید برای خودمان است تا بتوانیم یک تاپیک را بوکمارک کنیم تا بعداً راحت‌تر به آن دسترسی داشته باشیم. در حالت عادی جواب‌های داده شده به یک سوال بر حسب

این هم نمایی از serverfault:

The screenshot shows the ServerFault website interface. At the top, there's a navigation bar with links for 'Questions', 'Tags', 'Users', 'Badges', 'Unanswered', and an 'Ask Question' button. A search bar is also present. Below the navigation bar, there's a 'Recent Questions' section with a list of questions, each showing its title, votes, answers, views, tags, and the user who asked it. For example, the first question is 'What *nix based command can I use to find my external IP?' by Brad Gilbert, with 1 vote, 7 answers, and 123 views. To the right of the questions, there's a 'Greetings!' box with a welcome message and a link to 'about' and 'faq'. Below that, there's an 'Interesting Tags' section with a search bar and an 'Add' button. At the bottom right, there's a quote from William Durkin: 'Network resilience... It's the cherry on top. I definitely recommend using SQL Backup over native backups.'

Subversion •

VisualSVN •

Beyond Compare 3 •

لایه وب (Web Tier):

– ۲ دستگاه Lenovo ThinkServer RS110

– سی پی یو ۴ هسته‌ای 2.86 GHZ با 12MB کش L2

– هارد درایو های دیتاسنتر: 500GB

– رم: 8GB

– 500GB RAID 1 mirror array

• لایه دیتابیس:

– یک دستگاه Lenovo ThinkServer RD120

– سی پی یو ۸ هسته‌ای 2.5GHZ با 24MB کش L2

– رم: 48 GB

• یک سرور چهارم نیز برای superuser.com اضافه شد که

مجموعاً این سرورها برای superuser ، StackOverflow و

serverfault به کار می روند.

• هاستینگ: www.peakinternet.com

• برای سرچ سایت از قابلیت "SQL" Full Text Search

Server استفاده شد

درس‌هایی که گرفته شد (از زبان خود Joel و Jeff):

• اگر می‌توانید سرورها را پیکربندی کنید، به جای کرایه خودتان

خب، حالا به مشخصات فنی این سایت بپردازیم:

آمار سایت:

• حدود ۱۶ میلیون صفحه مشاهده در ماه

• ۳ میلیون مشاهده کننده ی مجزا در ماه (مقایسه: Facebook

۷۷ میلیون مشاهده کننده ی مجزا در ماه دارد)

• ۶ میلیون مشاهده در ماه

• ۸۶ درصد ترافیک از طرف گوگل می‌آید

• ۹ میلیون برنامه‌نویس فعال (۳۰ درصد) از StackOverflow

استفاده کرده‌اند

• شیوه‌های کسب درآمد: تبلیغات (البته از آن تبلیغاتی که کاربر

را اذیت کرده یا مزاحم خواندن صفحات شود!)، تبلیغات مربوط

به کار (آگهی استخدام. http://jobs.stackoverflow.com) ،

کنفرانس‌های DevDays ، استفاده مجدد از بستر نرم افزاری در

دیگر سایت‌ها مثل Serverfault و SuperUser) ...

پلتفرم و فن آوری‌های به کار رفته:

• Microsoft ASP.NET MVC

• SQL Server 2008

• C#

• Visual Studio 2008 Team Suite

• JQuery

• LINQ to SQL

VPN دارید. در صورت افزودن یک وب سرور دیگر، نیاز به یک اپلاینس load balancing دارید. قیمت این دستگاه‌ها به راحتی حدود ۲ برابر چندین سرور می‌شود

- مترجم: Scale Out یعنی افزودن ماشین‌های بسیار برای موازی سازی و در نهایت افزایش پرفورمنس برای انجام کاری؛ Scale Up یعنی قدرتمند کردن یک سرور (مثلاً افزودن رم). حتماً این پست را بخوانید تا با یک مثال واقعی همراه با اعداد آشنا شوید: <http://www.codinghorror.com/blog/archives/0۰۱۲۷۹.html>.

اگر از برنامه‌های این سورس استفاده می‌کنید Scale out راه حل خوبی است.

در غیر این صورت Scale Up یعنی پرداختن هزینه کم‌تر برای لایسنس نرم افزارها (و در عوض پرداختن هزینه بیشتر برای سخت افزار) و Scale out یعنی هزینه کم‌تر برای سخت افزار و هزینه بیشتر برای نرم افزار.

- RAID-10 برای شرایط کاری read/write سنگین دیتابیس عالی است

- منطق برنامه و دیتابیس را از هم جدا کنید تا هر کدام جدا از دیگری scale شوند. دیتابیس‌ها Scale up و برنامه‌ها Scale out می‌شوند.

- مشکل Scale up، فقدان افزونگی است. یک کلاستر قابلیت اطمینان را بالا می‌برد اما وقتی ماشین‌ها گران باشند، روش بسیار گرانی خواهد بود

- برنامه‌های کمی می‌توانند همراه با افزایش تعداد پردازنده‌ها به صورت خطی scale شوند. Lock ها باعث کاهش پرفورمنس می‌شوند

- در صورت افزودن دیتابیس سرورها، هزینه لایسنس SQL Server وحشتناک می‌شود.

سرور بخرید. کرایه کردن دو مشکل دارد: اول این که قیمت افزایش حافظه و دیسک خیلی بالا است و دوم این که معمولاً آن‌ها (صاحبان سرور) قادر به مدیریت سرورها نیستند.

- اگر ابتدای کار زیاد هزینه کنید در دراز مدت به نفعتان است تا این که هر ماه خرج‌های تکراری و اضافی داشته باشید

- تمام درایورهای شبکه‌تان را به روز کنید (حداقل کارایی ۲ برابر می‌شود)

- در صورت استفاده از GB 48 رم مجبورید از ویندوز سرور Enterprise Edition استفاده کنید

- حافظه خیلی ارزان است؛ بیشترین حافظه ممکن را بخرید
- ابتدا ما یک بخش کلیدی و مهم طرح دیتابیس Wikipedia را کپی کردیم که بعداً معلوم شد کار اشتباهی کرده‌ایم زیرا نیاز به کلی refactoring داشت. (باید از شر Join های اضافی در بسیاری از کوئری‌ها راحت می‌شدیم).

از دیتابیس‌های چندین ترابایتی غول این درس را یاد گرفتیم: از Join استفاده نکنیم. تقریباً تمام دیتابیس سایت در رم قرار دارد اما با این وجود باز هم عمل join هزینه بالایی دارد.

- سرعت سی پی یو به طرز شگفت‌آوری برای سرور دیتابیس مهم است. وقتی از GHZ 1.86 به GHZ 2.5 و سپس به 3.5 GHZ ارتقا دادیم تقریباً یک بهبود خطی را در زمان‌های کوئری‌ها مشاهده کردیم. (حالت استثنا شامل کوئری‌هایی است که در حافظه نمی‌گنجند)

- ۹۰ درصد مواقع دیتابیس گلوگاه است
- در سرورهای با حجم پایین تعیین کننده‌ی اصلی قیمت فضای rack، توان مصرفی برق، پهنای باند، سرورها یا نرم افزار نیستند بلکه "تجهیزات شبکه" است.

شما نیاز به یک شبکه گیبابیت بین لایه‌ی دیتابیس و وب دارید. بین cloud و وب سرورتان نیاز به فایروال، روتر و دستگاه‌های

اطلاعات بیشتر در مورد معماری StackOverflow

<http://blog.stackoverflow.com/2008/09/what-was-stack-overflow-built-with/>

http://www.youtube.com/watch?v=NWHfY_IvKIQ

<http://www.codinghorror.com/blog/archives/001279.html>

http://www.sqlserverpedia.com/wiki/Understanding_the_StackOverflow_Database_Schema

<http://blog.stackoverflow.com/2008/12/server-hosting-rent-vs-buy/>

<http://www.codinghorror.com/blog/archives/001283.html>

Reflector

معرفی ابزار

نویسنده: مهدی عسگری



از کاربردهای این ابزار می‌توان به این موارد اشاره کرد:

- مهندسی معکوس (مطالعه کد دیگران و پی بردن به روش‌ها و الگوریتم‌های به کار برده شده)

- مطالعه کد **.NET Framework** و آگاهی بیشتر از پلتفرم زیرین خود (مثلاً من نمی‌دانستم که با **Close** کردن یک **StreamWriter** بافر آن **Flush** هم می‌شود تا این که کد مربوط به این متد را با **Reflector** مشاهده کردم)

- تسریع در یادگیری زبان‌های مختلف (فرض کنید می‌خواهید **Visual Basic** یاد بگیرید؛ کدهای مختلفی را با سی شارپ نوشته و کامپایل می‌کنید، سپس با این ابزار کد **VB** آن را مشاهده می‌کنید. به طور پیش فرض زبان‌های **C#**، **Visual Basic**، **IL**، **Delphi**، **Chrome**، و **++MC** در این نرم افزار وجود دارند و زبان‌های دیگر نیز قابل افزودن هستند)

یک ویژگی این ابزار، قابلیت گسترش و افزودن قابلیت‌های مختلف به آن از طریق افزونه‌های مختلف است. مثلاً افزونه مربوط به **decompiler** زبان **IronPython**، افزونه‌ای برای مشاهده تفاوت بین اسمبلی‌ها و ... (افزونه‌های زیادی برای این ابزار وجود دارد که از آدرس www.codeplex.com/reflectoraddins قابل دانلود هستند)

کار با این ابزار نیاز به هیچ دانش پیشینی ندارد، فقط کافی است با فشردن **Ctrl + O** یک اسمبلی ایجاد شده توسط یک زبان دات نت را با آن باز کرده و کد متدهای آن را ببینید!

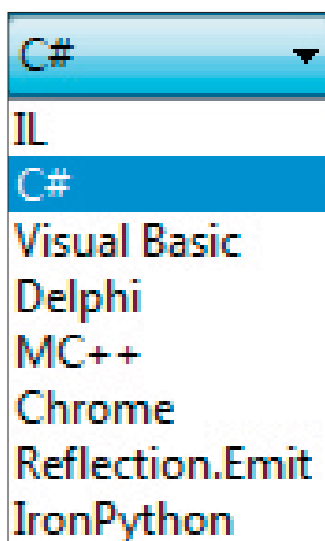
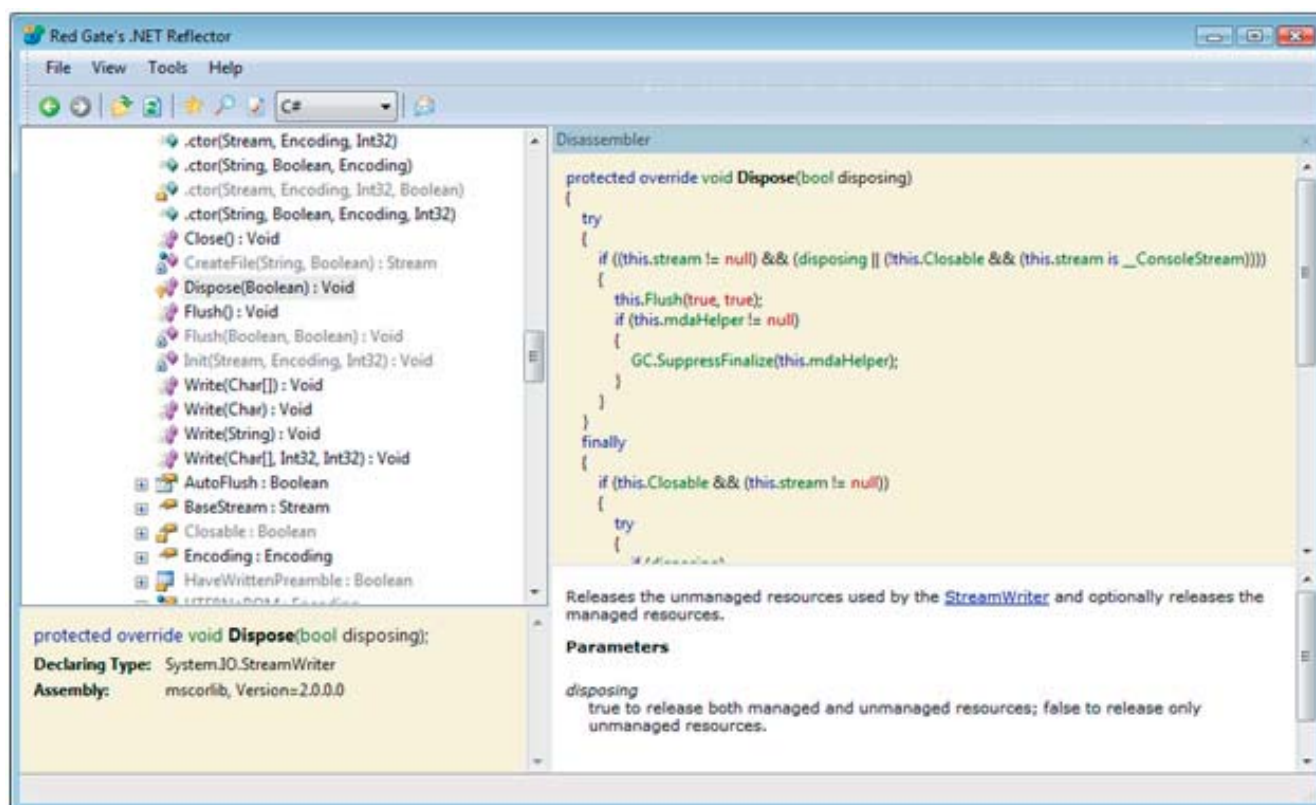
عکس‌هایی از محیط این ابزار:

برای این شماره پارتی‌بازی کرده و ابزاری برای برنامه‌نویسان **.NET** معرفی می‌کنم. البته اکثر برنامه‌نویسان حرفه‌ای با این ابزار آشنا هستند.

امروز با **Reflector** آشنا می‌شویم. این ابزار یک **Decompiler** است یعنی کار برعکس یک کامپایلر را انجام می‌دهد. به زبان ساده: اگر شما یک فایل اجرایی (**exe** یا **dll**) ایجاد شده توسط یکی از زبان‌های دات نت (مثلاً سی‌شارپ) را با این نرم افزار باز کنید، می‌توانید کدی را که این برنامه با آن نوشته شده، به زبان اصلی (سی‌شارپ، **VB**، ...) مشاهده کنید (یک نوع مهندسی معکوس) قدیمی‌ترها و نیز برنامه‌نویسان زبان‌های **native** (مثل **C++**) می‌دانند که این کار در زبان‌های پیشین بسیار سخت (اگر نه غیر ممکن) بود و ما فقط قادر بودیم کد اسمبلی فایل اجرایی را مشاهده کنیم (به همین علت به ابزاری که این کار را می‌کرد **disassembler** گفته می‌شد/ می‌شود). در محیط‌هایی مثل دات نت عمل کامپایل و اجرای یک برنامه در دو مرحله انجام می‌شود. کامپایلر فقط یک کد میانی تولید می‌کند (که در دات نت به آن **IL** یا **Intermediate Language** گفته می‌شود) و عمل تبدیل این کد به زبان ماشین برعهده **runtime** است.

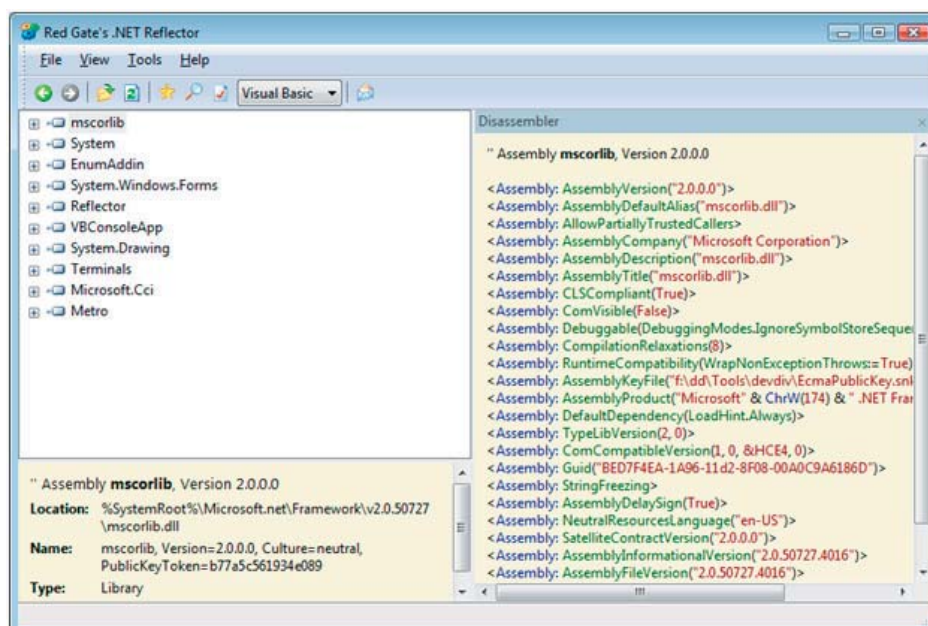
در دات نت علاوه بر **IL** داده‌هایی نیز تولید شده و در فایل خروجی گنجانده می‌شوند که به آن **metadata** گفته می‌شود. **Metadata** حاوی اطلاعاتی درباره کد و داده‌های موجود در فایل نهایی است. ابزاری مانند **Reflector** نیز از این اطلاعات استفاده کرده و کد منبع یک فایل اجرایی را به ما نشان می‌دهد. البته خود **Visual Studio** نیز ابزاری مشابه با نام **ILDisassembler** دارد، منتها با این محدودیت که فقط کد **IL** و **metadata** می‌فایل‌ها را نشان می‌دهد. تفاوت **Reflector** در این است که علاوه بر این کار، می‌تواند کد **IL** را به زبان‌های سطح بالاتر تبدیل کند که کار ما را در خواندن کد راحت تر می‌کند.

این ابزار فوق العاده که توسط **Lutz Roeder** نوشته شده تا جایی که به یاد دارم از نسخه‌های بتای **.NET** شروع به کار کرده و تا به امروز که آخرین نسخه آن **5.1.5.0** است توسعه آن ادامه دارد. البته سال قبل این ابزار به شرکت **Red Gate** فروخته شد (البته قول داده‌اند که این ابزار همیشه رایگان خواهد بود و تا امروز نیز همینطور بوده)

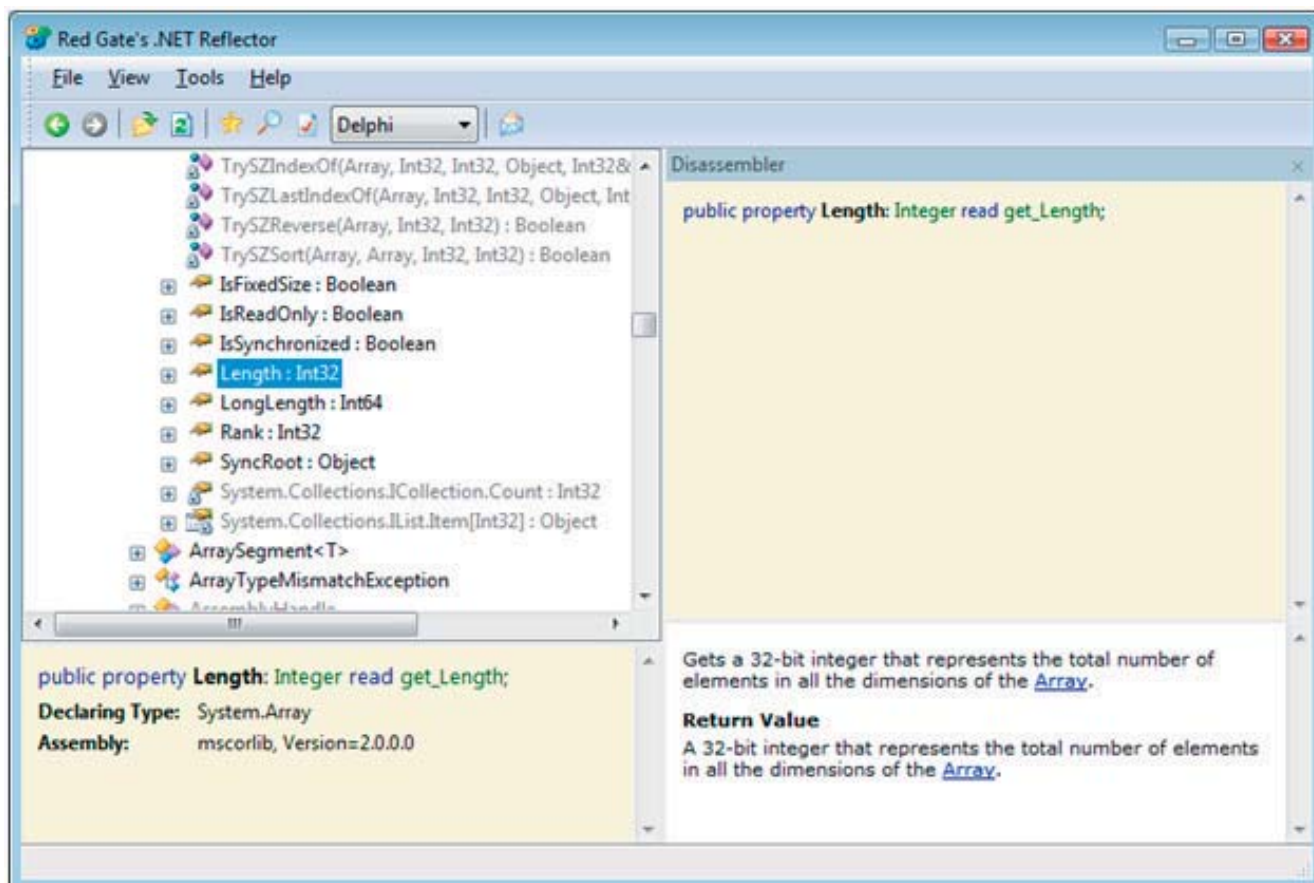


صفحه به چهار قسمت تبدیل شده: در ربع اول (بالا، چپ) لیست درختی اسمبلی‌ها، کلاس‌ها، متدها، فایل‌ها و ... وجود دارد. در قسمت بالا راست، کد متد انتخاب شده مشاهده می‌شود (در مثال بالا زبان C# در نوار ابزار انتخاب شده)

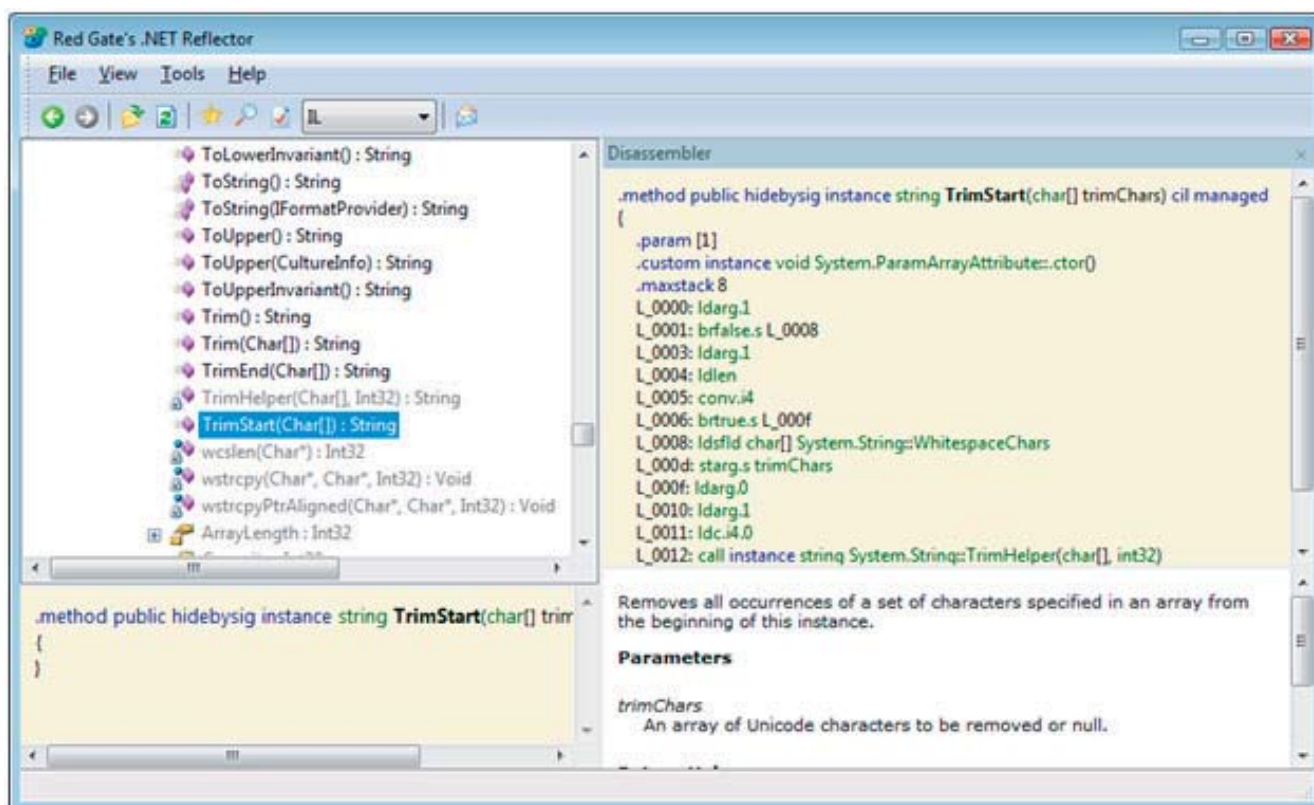
در قسمت پایین چپ، امضای کامل متد به همراه نوع و اسمبلی‌ای که متعلق به آن است دیده می‌شود (مثلاً در اینجا تابع Dispose از کلاس StreamWriter است) و در قسمت پایین راست نیز مستندات مربوط به آیت‌م انتخاب شده مشاهده می‌شود.



در شکل روبه‌رو نیز زبان‌های موجود مشاهده می‌شوند (زبان IronPython به طور پیش فرض در این ابزار وجود ندارد و از طریق نصب یک افزونه اضافه شده است. البته برای نصب افزونه فقط کپی کردن یک فایل dll به فولدر Reflector کافی است!)



اطلاعات مربوط به پراپرتی **Length** یک **Array** (به زبان Delphi) (این فقط اعلان پراپرتی است و کد آن توسط متد **get_Length** پیاده‌سازی شده است)



کد تابع **TrimStart** کلاس **String** به زبان IL

جامعه برنامه نویسان فارسی زبان برگزار می کند:

- « دوره عملی "فروشگاه الکترونیک (eShop) با استفاده از C# و ASP.NET
- « دوره آموزشی برنامه نویسی پروژه Portal/CRM با استفاده از C# و ASP.NET
- « دوره کامل آموزشی برنامه نویسی Web با استفاده از PHP و MySQL
- « دوره کامل آموزشی برنامه نویسی Web با استفاده از ASP.NET و C#
- « دوره کامل آموزشی برنامه نویسی Web با استفاده از ASP.NET و VB.NET
- « دوره آموزشی کاربردی Prado Framework در PHP

جهت کسب اطلاعات بیشتر به آدرس www.barnamenevis.org مراجعه نمایید.

برگزاری دوره های آموزشی برنامه نویسی



نگاهی نزدیک تر به IP Telephony

Sys-developer@Maxnet.ir



با حرکت خود جنبشی را در این صنعت کاملاً جاافتاده در سایر کشورها، در کشور عزیزمان هم ایجاد نموده و خود را در این زمینه نیز به دهکده جهانی اینترنتی (که پایه این صنعت است) پیوند زنند. ناگفته پیداست که این امر کاملاً دوسویه بوده و تأثیری مستقیم بر تمام جامعه VoIP کاران و ازجمله نویسنده، خواهد گذاشت و با اشتراک گذاری تجربیات؛ نواقص در این زمینه تا حد امکان مرتفع می گردند.

۲. پیش نیازها:

در این مقاله فرض بر این است که خواننده با حداقل مفاهیم مخابراتی از جمله PSTN، PBX، VoIP و خطوط آنالوگ و دیجیتال و سیگنالینگ و ...؛ همچنین مفاهیم سورس باز و نرم افزارهای انتشار یافته با این مجوز از جمله Asterisk/Trixbox آشنایی مقدماتی دارد؛ درضمن به علت سورس باز بودن منابع، فرض بر این است که خواننده یکی از ویرایش های لینوکس و IP-PBX دلخواه را روی آن نصب نموده است (توجه نمایید که هر دو Asterisk/Trixbox دارای نسخه های Bootable سفارشی شده لینوکس - اکثراً Red Hat با کرنل 2.4 یا 2.6 می باشند). در ادامه جهت پیشبرد سناریو برخی از مفاهیم مرتبط با پروتکل های SIP، IAX و RTP، TCP/IP نیز با فرض آشنایی ابتدایی خواننده مورد استفاده قرار می گیرند.



۱. مقدمه:

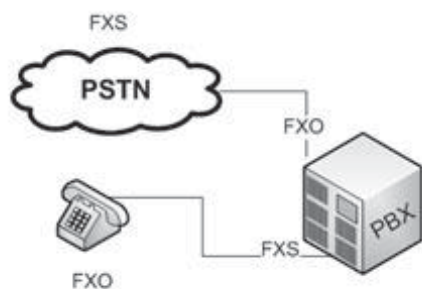
این مقاله پیرو نسخه قبلی آشنایی با VoIP و تلاشی در جهت تکمیل سناریوهای ارائه شده می باشد. با وقوف به این نکته که هم اکنون بسیاری از ارائه دهندگان سیستم های مبتنی بر VoIP در سرتاسر دنیا، هزینه بسیار اندکی (هزینه راه اندازی رایگان و اشتراک ناچیز) از کاربران دریافت کرده و اصلی ترین درآمد آن ها، هزینه سرویس های افزوده می باشد؛ دیگر نمی توان مباحث مرتبط با IP تلفنی را صنعتی انحصاری و پیچیده دانست و این امر مرهون سورس باز بودن و به اشتراک گذاری منابع اطلاعاتی پیشروان این صنعت انفجاری! می باشد.

نویسنده این مقاله، با علم به این مطلب و این که تجربیات و مستندات فراوان دیگران در این زمینه، ناخودآگاه هر محقق را به سمت جلو رهنمون می گردد، امیدوار است که خوانندگان گرامی با بهره گیری از مطالب گردآوری شده و تجربیات اندک این حقیر،

۳. جدول برخی اختصارات مرتبط با VoIP:

| مخفف | کامل | توضیح |
|------|----------------------------|--|
| ATA | Analogue Telephone Adapter | مبدل تلفن آنالوگ به تلفن مبتنی بر IP |
| FXO* | Foreign Exchange Office | درگاه RJ11 ای که همانند یک تلفن معمولی بوده و تجهیزات متصل به آن باید قادر به دریافت سیگنال زنگ و گوشی برداری-گذاری باشند. |
| FXS* | Foreign Exchange Station | درگاه RJ11 ای که خط آزاد به آن وصل می شود. در یک خط آنالوگ، FXS خط آزاد و توان مصرفی برای تلفن را فراهم می نماید. |

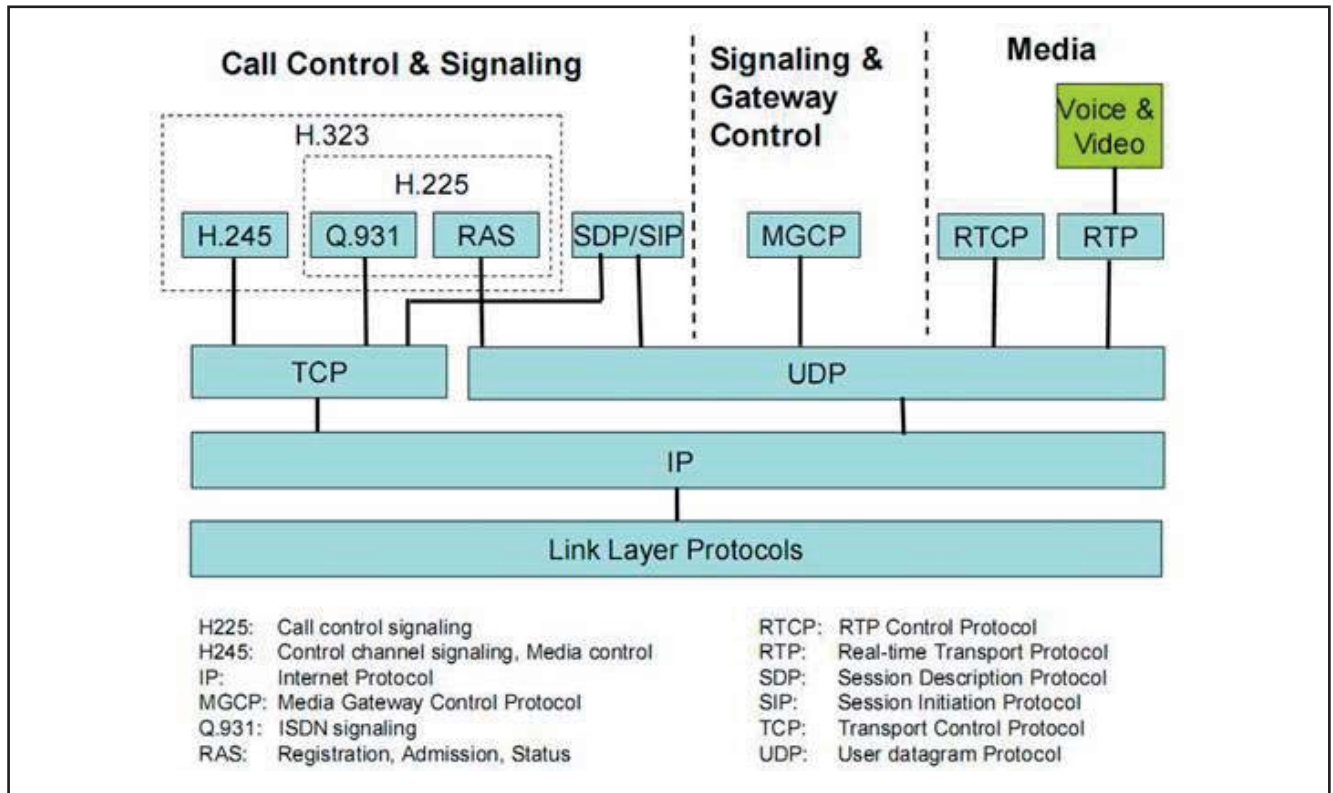
| مخفف | کامل | توضیح |
|---------------------|--|--|
| GSM | Global System for Mobile communication | شبکه مبتنی بر مدل لانه زنبوری که عمومی ترین استاندارد جهت تجهیزات موبایل می باشد. |
| IAX (IAX2) | Inter Asterisk eXchange protocol (version 2) | پروتکل اختصاصی Asterisk با RFC 5456 |
| IETF | Internet Engineering Task Force | کمیته استاندارد سازی اینترنت |
| ITU | International Telecommunications Union | کمیته بین المللی مخابرات |
| IVR | Interactive Voice Response | به سیستم های پاسخگوی خودکار تلفنی اطلاق می گردد. |
| NAT | Network Address Translator | به طور خلاصه به عنوان واسط بین یک شبکه محلی و اینترنت عمل می نماید : NAT به یک دستگاه مجاز در اینترنت اجازه می دهد که به صورت یک روتر عمل نماید. |
| PBX (PABX) * | Private (Automatic) Branch Exchange | به طور عمومی به مراکز تلفن محلی اطلاق می گردد. اصلی ترین وظیفه PBX قرار گرفتن بین یک یا چند خط تلفن و تعدادی از کاربران و تقسیم تماس های دوطرفه بین دو طرف است. |
| PCM | Pulse Code Modulation | نمایش دیجیتالی یک سیگنال آنالوگ به صورت بازه های 0 و 1 ؛ همچنین به عنوان استاندارد در صوت و تصویر دیجیتال نیز استفاده می گردد. |
| PSTN* | Public Switched Telephone Network | شبکه عمومی مخابرات که به عنوان شبکه تلفن ثابت نیز از آن یاد می شود. |
| QoS | Quality of Service | در شبکه های مبتنی بر بسته (Packet-switched)، به مکانیزم کنترل ذخیره منابع، جهت دستیابی به کیفیت سرویس مناسب اطلاق می گردد. |
| RFC | Request For Comment | یادداشت های منتشر شده IETF که روش، رفتار، تحقیقات و نوآوری های تصویب شده در زمینه اینترنت و سیستم های متصل به آن را توضیح می دهند و دارای شماره منحصر بفردی می باشند. |
| RTP | Real-time Transport Protocol | استاندارد فرمت بسته های صوتی و تصویری در اینترنت RFC 3550 |
| SCCP | Skinny Call Control Protocol | پروتکل اختصاصی سیسکو جهت کنترل ترمینال شبکه بین ایستگاه های کاری و CallManager های سیسکو می باشد که ابتدا توسط شرکت سلسیوس ابداع گردید و اکنون مالک و معرف آن سیسکو می باشد. |
| SIP | Session Initiation Protocol | پروتکل سیگنالینگ صوتی مبتنی بر VoIP که مفصلاً در مورد آن صحبت خواهد شد. |
| SS7 | Signaling System 7 | مجموعه ای از پروتکل های سیگنالینگ تلفنی که برای راه اندازی اکثر PSTN ها بکار گرفته می شود. |



* برای درک بهتر مفاهیم می توان ارتباط موارد علامت گذاری شده با * شکل روبرو را مشاهده نمایید، در این شکل نحوه تعامل، PBX، FXS، FXO و تلفن معمولی مشاهده می شود.

۴. سیگنالینگ در IP Telephony :

پشته پروتکل‌های مرتبط با VoIP را می‌توان در شکل زیر دسته‌بندی نمود؛ ولی در این گفتار، تمرکز اصلی روی دو پروتکل VoIP که در پیاده سازی IP-PBX ما نقش اساسی داشته و وظیفه سیگنالینگ را نیز بر عهده دارند، خواهد بود: SIP و IAX2. به سایر پروتکل‌ها در صورت ضرورت نگاهی گذرا جهت آشنایی خواهیم انداخت. در مورد سیگنالینگ^۱ VoIP هم باید دانست که فلسفه مشابهی با سیگنالینگ PSTN معمولی دارد؛ بدین معنا که مکالمات و سیگنال‌ها آشکارا متمایز هستند، توضیحات بیشتر در قسمت‌های بعد به تفصیل خواهند آمد.



۴-۱. H.323

این پروتکل توسط ITU-T^۲ استاندارد شده و آخرین نسخه عمومی آن در زمان تحریر این مقاله نسخه چهارم می‌باشد. و پروتکلی P2P می‌باشد که از ترمینال‌های ارتباطی روی شبکه‌های مبتنی بر بسته استفاده می‌کند.

همان‌طور که از شکل برمی‌آید، H.323 شامل پروتکل‌های زیر می‌شود:

- سیگنالینگ تماس: (Q.931) H.255
- کنترل رسانه ای: H.245
- رسانه انتقال: RTP, RTCP
- رمزگذاری صوتی: G.711, G.722, G.723, G.728, G.729
- رمزگذاری تصویری: H.261, H.263
- H.323 پروتکل بسیار سنگینی است و می‌توان گفت که همه چیز

را در مکانیزم ارتباطی چندرسانه‌ای پوشش می‌دهد، این پروتکل در ارتباطات چندرسانه‌ای و ویدئوکنفرانس‌های آنلاین بسیار مورد استفاده و قدرتمند است، و یکی از فراگیرترین پروتکل‌ها در زمینه VoIP است ولی باید در نظر داشت که پروتکلی کاملاً باینری بوده و کاربردهای وسیعی دارد و مختص VoIP نیست؛ بنابراین رفع اشکال در آن بسیار زمان‌بر و پیچیده خواهد بود. به علت گستردگی زیاد این پروتکل بحث در مورد آن را به فرصت‌های آتی واگذار می‌کنیم.

۴-۲. SIP (Session Initiation Protocol)

SIP پروتکلی نسبتاً ساده است که مشخصه‌های مشابه زیادی با پروتکل‌های HTTP و SMTP دارد و این باعث می‌گردد که اشکال‌زدایی از آن بسیار ساده شود به همین دلیل اکثراً از آن به عنوان یکی از "پروتکل‌های ساده تلفنی"^۳ یاد می‌شود. این پروتکل

۱- در زمینه تماس‌های تلفنی بطور خلاصه منظور از سیگنالینگ؛ تشخیص اشغال خط، صدای زنگ و گوشی برداری طرف مقابل و ... می‌باشد.

۲- International Telecommunication Union: اتحادیه جهانی مخابرات؛ رک. به: <http://www.itu.int/ITU-T>

۳- Lightweight Telephony Protocol یا LTP: پروتکل‌هایی که دارای انعطاف پذیری زیاد و در عین حال سادگی می‌باشند؛ از جمله سایر چنین پروتکل‌هایی می‌توان به NAT, UDP اشاره نمود. رک. به: <http://www.lightweighttelephony.org> و به عنوان نمونه: <http://www.packetmobile.com/ltp.html>

این امر مستثنی نیست. NAT دشمن بزرگی برای RTP محسوب می‌شود. شبکه‌های مبتنی بر NAT شامل دسته‌ای از کامپیوترها هستند که برای ارتباط با خارج از شبکه‌شان از یک آدرس عمومی اینترنت استفاده می‌کنند. درواقع هزینه‌ای که شبکه مبتنی بر NAT بابت آسان شدن ارتباطش با اینترنت می‌پردازد این است که کامپیوترهای داخل این شبکه را از حالت routable بودن (دسترسی کامل از طریق IP) خارج می‌کند. مهم‌ترین اشکالی که این پروتکل برای VoIP ایجاد می‌کند تحت عنوان "ارتباط یک طرفه" از آن یاد می‌شود، دلیل آن هم همان‌طور که ذکر گردید مجزا بودن بسته‌های ارسالی / دریافتی می‌باشد؛ "شماره گیرنده" داخل NAT قادر به ارسال بسته‌های صوتی خود می‌باشد در صورتی که بسته‌های صوتی بازگشتی به علت ماهیت NAT بلوکه می‌شوند.

۳-۴. IAX (Inter-Asterisk eXchange)

این پروتکل جدیدتر از پروتکل‌های SIP و H.323 می‌باشد و نسبت به آن‌ها به تازگی به جمع پروتکل‌های VoIP پیوسته است. IAX پروتکل اختصاصی Asterisk^۵ می‌باشد و بالطبع به وسعت پروتکل‌های قبل از آن استفاده نمی‌گردد ولی سازگاری بیشتر آن با NAT نسبت به دو پروتکل قبل باعث استقبال عمومی زیادی نسبت به آن شده است. هم‌اکنون عموماً نسخه دوم آن بنام IAX2^۶ به عنوان پروتکل غالب IAX شناخته می‌شود. از پروتکل UDP (غالباً روی پورت 4569) برای ارتباط بین دو طرف و انتقال سیگنالینگ و داده استفاده می‌کند. برخلاف SIP که از یک زوج رشته‌های مجزا (یکی برای سیگنالینگ و دیگری صوت) استفاده می‌کند؛ IAX2 یک رشته منفرد از داده‌ها را برای ارتباط با end-point ها برقرار می‌کند. در IAX2 هر دوی داده و سیگنالینگ روی یک کانال انتقال می‌یابند (in-bound) در تقابل با SIP که همان‌طور که ذکر شد بسته‌ها بصورت (out-bound) با RTP منتقل می‌شوند. از طرف دیگر IAX2 اجازه انتقال چندین تماس مجزا در مجموعه واحدی از بسته‌های IP را می‌دهد، که این مکانیزم به عنوان "Trunking" شناخته می‌شود. بدینوسیله IAX2 قادر به ذخیره پهنای باند می‌شود. برای درک این مفهوم فرض کنید شما پنج نامه به مقاصد متفاوت دارید؛ روش اول استفاده از پنج پاکت مجزا برای هر مقصد می‌باشد ولی راه دیگر این است که آدرس اولین مقصد را روی یک پاکت نوشته و در ابتدای هر نامه مقصد بعدی را مشخص نمایید، بدین ترتیب می‌توانید تمام پنج نامه (تماس‌ها) را درون یک پاکت (بسته IP) قرار دهید.

از هر دو پروتکل TCP^۴ و UDP^۵ می‌تواند بهره‌بردار ولی اکثراً از UDP استفاده می‌کند. سرویس‌دهنده‌های فراوانی از این پروتکل جهت انتقال سرویس‌های خود استفاده می‌کنند. این پروتکل توسط سازمان IETF برای پشتیبانی از برخی توابع SS7 توسعه یافته است. به صورت خلاصه SIP در شبکه‌های ارتباطی مبتنی بر IP مسئولیت برقراری مکالمه و سیگنالینگ را در سه مرحله زیر انجام می‌دهد:

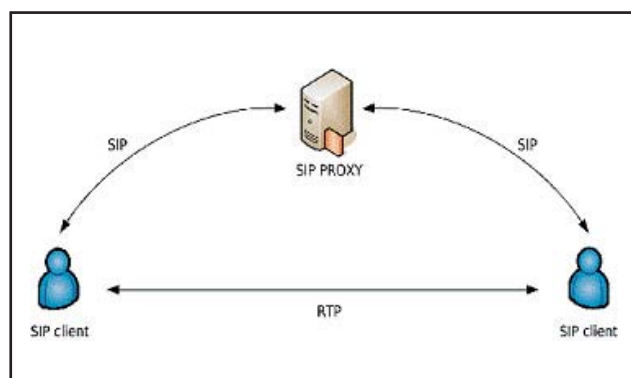
۱. انجام مراحل شناسایی و احراز هویت

۲. تهیه مناسبات کیفیت تماس تلفنی

۳. مدیریت آدرس IP و شماره پورت به کار رفته هنگام ارسال "مکالمات صوتی"

۴-۲. SIP Proxy Servers

اگرچه دو وسیله مبتنی بر SIP (IP Phone) می‌توانند مستقیماً با هم ارتباط داشته باشند ولی SIP برخی عناصر دیگر را برای آسان نمودن ارتباط تماس تلفنی اضافه نموده که از آن‌ها به عنوان SIP Proxy Server یاد می‌شود. در تلفن اینترنتی شما می‌توانید تلفنتان را با خودتان به هر جای دنیا که خواستید ببرید! به عبارت دیگر شما به محل فیزیکی‌تان گره نخورده‌اید. SIP Proxy Server طی فرآیندی که Registration نام دارد می‌فهمد که کاربر موردنظر در چه مکانی قرار گرفته و ارتباط را برقرار می‌کند.



۴-۲-۲. NAT و RTP

در اینترنت، تماس‌های تلفنی SIP رشته‌ای از بسته‌های کوچک هستند که توسط پروتکل دیگری بنام RTP حمل می‌شوند. RTP فرمت استاندارد برای حمل بسته‌های صوتی و تصویری در اینترنت تعریف می‌کند که در آن تماس‌های معمولی در بسته‌های ارتباطی به دو بخش مجزا تقسیم می‌شوند، که پروتکل SIP هم از

۴- Transmission Control Protocol از پروتکل‌های اساسی اینترنت، رک. به: http://en.wikipedia.org/wiki/Transmission_Control_Protocol

۵- User Datagram Protocol یکی دیگر از پروتکل‌های اصلی اینترنت، رک. به: http://en.wikipedia.org/wiki/User_Datagram_Protocol

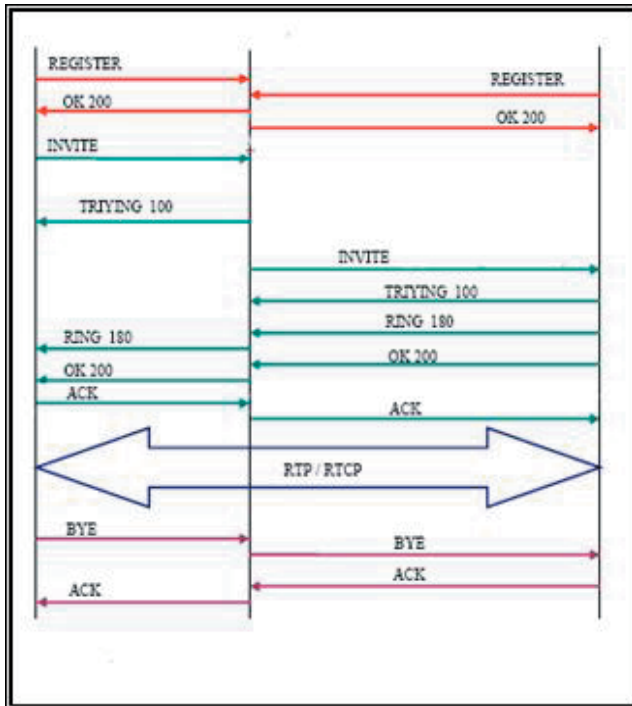
۶- نرم‌افزاری پیشرو در صنعت پیاده سازی PBX (مرکز سانترال تلفن) که همزمان قابلیت سرویس دهی به شبکه‌های PSTN و VoIP را دارا می‌باشد و مبدع آن Mark Spencer

از شرکت Digium در سال ۱۹۹۹ می‌باشد؛ نام آن نیز برگرفته از نام لاتین "ستاره کوچک" می‌باشد. رک. به: <http://www.asterisk.org>

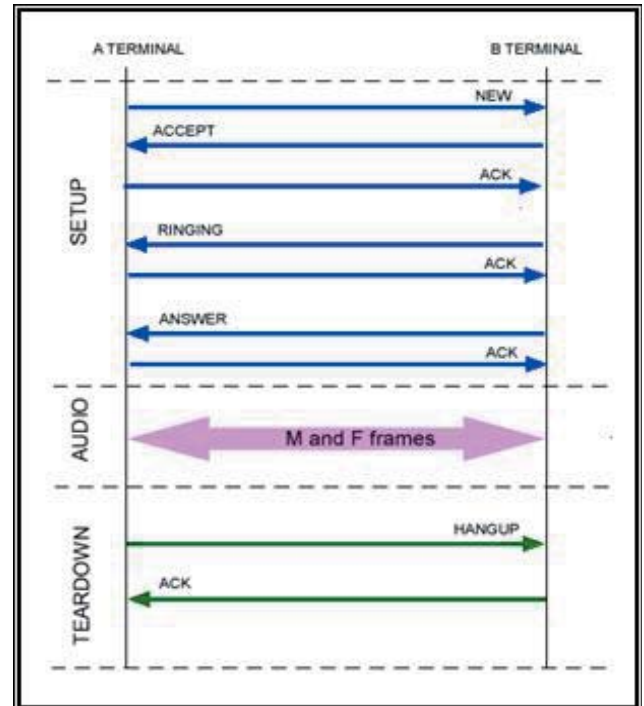
۷- IAX2 پروتکلی با سربر و پهنای باند کمتر است که اجازه ارتباط چندین Asterisk PBX با همدیگر را می‌دهد. اضافه بار (Payload) تنها در چهار اکتال (۳۲ بیت) با سربر هدر فرستاده می‌شود. البته هدرهایی با سربر بیشتر با ۱۲ اکتال برای کنترل توابع و برخی بسته‌های اضافه نیز استفاده می‌شوند. (تقریباً هر دقیقه یکبار)

نحوه تراکشی‌های ارتباطی :

تعاملات ارتباطی SIP



تعاملات ارتباطی IAX



۵. مقایسه اجمالی پروتکل‌های VoIP (SIP و IAX)

قبل از ادامه بحث باید خاطر نشان کرد که با توجه به حجم مقاله حاضر، این مقایسه‌ها را نمی‌توان به صورت مرجع و کامل در نظر گرفت و هدف تنها آشنایی خواننده با تفاوت‌های اساسی پروتکل‌های جاری VoIP می‌باشد؛ هر کدام از این پروتکل‌ها نسبت به مورد استفاده، دارای مزایای خود می‌باشند و هیچکدام ذاتاً خارج از استفاده یا بهترین و ... نمی‌باشند، ولی به طور کلی از نظر فنی، پروتکل IAX2 حداقل از مزایای زیر نسبت به SIP بهره می‌برد :

- به حداقل رساندن استفاده از پهنای باند به ازای هر تماس
- فراهم نمودن پشتیبانی ذاتی از شبکه‌های NAT (استفاده آسان تر پشت دیواره‌های آتش)
- به حداقل رساندن استفاده از پهنای باند برای تماس‌های متعدد (بوسیله trunking)

۵-۱. از نظر پهنای باند مورد استفاده^۸:

| Codec | SIP | | IAX (Trunked) | |
|----------------|------------|----------------|---------------|----------------|
| | اولین تماس | تماس‌های اضافی | اولین تماس | تماس‌های اضافی |
| G.711 (64kbps) | 80kbps | 80kbps | 80kbps | 64kbps |
| G.726 (32kbps) | 48kbps | 48kbps | 46kbps | 32kbps |
| G.729 (8kbps) | 24kbps | 24kbps | 23kbps | 8kbps |
| G.722 (64kbps) | 80kbps | 80kbps | 80kbps | 64kbps |
| GSM (13kbps) | 29kbps | 29kbps | 28kbps | 13kbps |

پهنای باند ذکر شده شامل هدر IP و فقط برای یک طرف مکالمه در نظر گرفته شده است. برای بدست آوردن مقدار کامل مقادیر نمایش داده شده را دو برابر نمایید.

۸- برگرفته از http://www.astricon.net/۲۰۰۸/glendale/web/presentations/IAX_to_Carriers_Presentation.pdf

۵-۲. از نظر سازگاری با NAT:

پروتکل SIP در انتقال سیگنالینگ و داده روی NAT معمولاً دچار مشکلات عدم سازگاری می‌گردد که علت آن همانطور که پیش تر ذکر شد، به کارگیری پروتکل‌های متفاوتی برای انتقال می‌باشد. SIP معمولاً نیاز به استفاده یک STUN Server^۹ برای حل مشکل مزبور می‌باشد.

۵-۳. پروسه استانداردسازی کاربردی:

استانداردهای SIP مدتها پیش توسط IETF به تصویب رسیده‌اند و بصورت گسترده در حال استفاده می‌باشند ولی IAX هنوز در حال استانداردسازی می‌باشد و به همین دلیل خیلی از دستگاه‌ها

هنوز قادر به کار با IAX نبوده و حداقل تا هم اکنون تنوع نرم افزارهای سازگار با SIP بسیار بیشتر و گسترده تر می‌باشد.

۶. Codec های صوتی:

بطور خلاصه یک Codec صوتی، یک سیگنال صوتی را دریافت کرده و آنرا به فرمت باینری (صفر و یک) تبدیل می‌کند. Codec ای بهتر است که با پهنای باند معادل با دیگری، قادر به تهیه کیفیت مطلوب‌تر باشد. البته باید گفت که ممکن است متراکم سازی بیشتر، باعث اختلال بیشتری هم بشود (کیفیت پایین‌تر). به هر ترتیب، Codec های استاندارد صوتی را می‌توان در جدول زیر^{۱۰} خلاصه نمود:

| استاندارد | نحوه متراکم سازی | پهنای باند | امتیاز MOS |
|-------------|--|---------------------|------------|
| G.711 | Pulse Code Modulation (PCM) | 64 Kbps | 4/1 |
| G.723.1 r53 | MultiPulse-MultiLevel Quantization (MP-MLQ) | 5/3 Kbps | 3/9 |
| G.723.1 r53 | Algebraic Code Excited Linear Predictive (ACELP) | 6/4 Kbps | 3/65 |
| G.726 | Adaptive Differential PCM (ADPCM) | 40, 32, 24 و 16Kbps | 3/85 |
| G.728 | Low Delay-Code Excited Linear Predictive (LD-CELP) | 64 Kbps | 3/61 |
| G.729 | Conjugate Structure-Algebraic Code Excited Linear Predictive (CS-CELP) | 64 Kbps | 3/92 , 3/7 |

* MOS: معادل میانگین ریاضی تمام امتیازات جداگانه می‌باشد، و میتواند بین ۱ (بدترین امتیاز) و ۵ (بهترین امتیاز) توسان نماید.

۷. تجهیزات الحاقی VoIP:

با توجه به اینکه هر سیستمی بدون ارتباط با دنیای خارج عملاً بلااستفاده می‌شود، و همچنین راه انداختن یک سیستم برای ارایه خدمات، به تهابی نمی‌تواند جوابگوی نیاز مشتریان و کارفرمایان باشد تجهیزات و ضروریات دیگری همراه با سناریوهای VoIP مطرح می‌شوند که نیاز حیاتی یک سیستم مبتنی بر VoIP می‌باشند. علاوه بر اجزاء و نیازمندی‌های اصلی VoIP موارد دیگری نیز در چنین سیستم‌هایی وجود دارد، که در ادامه به برخی از آنها اشاره خواهد شد.

۷-۱. کارت ها و تجهیزات رابط PSTN:

برای مسیریابی تماسهای یک ترمینال VoIP به یک شبکه PSTN

همانطوری که از جدول بر می‌آید، با توجه به معماری سیستم و اهمیت فاکتورهای موردنظر توسعه دهنده سیستم (پهنای باند، سرعت انتقال و کیفیت انتقال) از یکی از این Codec ها استفاده می‌گردد. یک مدار دیجیتالی عمومی PSTN عموماً از Codec معروف PCM استفاده می‌کند. PCM یک Codec با کیفیت بالا است ولی نیاز به پهنای باند 64kbps دارد، و از دو استاندارد a-law (آمریکا و برخی دیگر کشورها مانند ایران) و μ -law (اروپا) استفاده می‌کند (این استانداردها به ترتیب G711a و G711u نیز شناخته می‌شوند). G.729 در کاربردهای VoIP، Codec خوبی است (به علت استفاده از پهنای باند کم)، ولی باید در نظر داشت که این Codec نیاز به اخذ پروانه جهت کاربرد تجاری دارد^{۱۱}.

۹- Simple Traversal of UDP through NATs، ر.ک. به <http://www.ietf.org/rfc/rfc۵۳۸۹.txt> و <http://www.voip-info.org/wiki-STUN>

۱۰- برگرفته از <http://www-ee.uta.edu/online/wang/VoIP.pdf>

۱۱- این Codec توسط کنسرسیومی از شرکتهای: France Telecom, Mitsubishi Electric Corporation, Nippon Telegraph and Telephone Corporation (NTT) و Université de Sherbrooke توسعه یافته است. قیمت این Codec حدوداً ۱۰ دلار ایالات متحده است.

۷-۱-۲. تأخیر (Latency):

این عامل معادل مدت زمانی است که طول می‌کشد تا یک بسته داده از یک نقطه معین به دیگری منتقل شود. برای افزایش کیفیت تماس‌های VoIP باید تلاش شود با دادن اولویت بیشتر به ترافیک صوتی، این تأخیر را به حداقل رساند. منظور از اولویت دادن به بسته‌های صوتی این است که داده‌های صوتی باید در صف بسته‌های انتقالی، به موقعیت بهتری جهش نمایند. به عنوان مثال، یک راه کاهش Latency قرار دادن PBX خودتان در گلوگاه شبکه (Saturated) می‌باشد.

۷-۲-۲. ناپایداری سیگنال (Jitter):

در VoIP، Jitter به ناپایداری در زمان رسیدن بسته‌ها به علت ازدحام شبکه، رانش زمانی (Timing drift) و یا تغییرات مسیریابی اطلاق می‌گردد. با داشتن یک بافر Jitter می‌توان این اختلال را مدیریت کرده و اثرات جانبی آنرا کم کرد. ایده اصلی بافر Jitter، بهبود دادن کیفیت تماس با قرار دادن عمدی مقداری تأخیر در پخش مکالمه می‌باشد، که به این طریق مطمئن می‌شویم که بسته‌های کند (Lazy packets) نیز خواهند رسید (البته نه به مقداری که همزمانی مکالمه زیرسوال برود).

در تجهیزات VoIP گزینه‌ای برای Jitter Buffer در اختیار کاربر قرار می‌گیرد که محیط مشترکی برای بسته‌های صوتی می‌باشد که در آنجا جمع‌آوری و ذخیره می‌شوند، و در فاصله‌های زمانی متناسب به پردازشگر صوتی فرستاده می‌شوند. Jitter Buffer که در محل انتهای اتصال صوتی دریافتی، قرار می‌گیرد، عمداً مقداری تأخیر در بسته‌های رسیده ایجاد کرده و بدینوسیله طرف پشت خط اتصالی واضح ولی با مقداری شکستگی صوتی (sound distortion) دریافت می‌نماید.

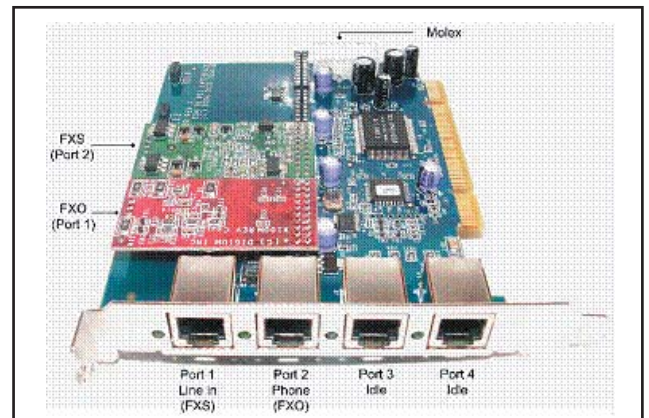
به طور کلی دو نوع Jitter Buffer وجود دارد؛ اولی Jitter Buffer ایستا، که به صورت سخت‌افزاری توسط سازنده تعبیه شده است. Jitter Buffer نوع بعدی نرم‌افزاری بوده و قابل تنظیم توسط کاربر می‌باشد. مقدار عمومی که برای این نوع بافر تعیین می‌شود ۱۰۰ میلی ثانیه می‌باشد، ولی شما می‌توانید برای داشتن نویز (خش) کمتر و بالا رفتن کیفیت مکالمه این بافر را افزایش دهید. ولی همیشه مراقب باشید که این افزایش یک تأخیر عمومی در مکالمات ایجاد می‌نماید.

۸. آستین‌ها بالا! یک سناریو عملی :

۸-۱. سناریو موردنظر

در این سناریو ما قصد داریم یک PBX برای چهار سازمان مجزای فرضی، در یک محدوده مرکز روستایی را راه‌اندازی کنیم. پس از کامل شدن مراحل نصب، اعضاء هر سازمان قادر خواهند بود به صورت رایگان با مخابرات بین روستایی خود و سایر سازمان‌ها و در نهایت دنیای خارج ارتباط تلفنی برقرار نمایند. در جدول زیر اطلاعات چهار سازمان و تکنولوژی‌های متفاوتی که

نیاز به یک سخت‌افزار اختصاصی در PBX خود دارید. تجهیزات متنوع و گسترده‌ای به این منظور وجود دارند که نسبت به نیاز سیستم‌ها، انتخاب‌های متفاوتی را فراهم می‌نمایند؛ از رابط‌های چند خط معمولی آنالوگ^{۱۲} گرفته تا Gateway‌های تخصصی برای مسیریابی خطوط متعدد دیجیتال (یا E1) با تنظیمات Built-in و Asterisk و... برای نمونه می‌توان از تجهیزات برندهای معتبری همچون CISCO، LUCENT، PATTON، ALCATEL و... که با تجهیزات تخصصی در این زمینه قادر به تامین همزمان سرویس‌های VoIP برای ده‌ها خط E1 و مدل‌های متفاوت Routing برای هریک از این خطوط در یک سخت‌افزار را مجتمع دارند، ولی به علت تمرکز مقاله روی نرم‌افزار از ارائه توضیحات بیشتر در این زمینه منصرف می‌شویم. برای شروع کار و بزرگ نشدن ابعاد پروژه، فرض می‌کنیم که برای ارتباط سازمان‌ها، کار ما با چند خط معمولی راه می‌افتد؛ با این فرض کارت Digium TDM400P می‌تواند یکی از مناسب‌ترین گزینه‌ها باشد؛ این کارت اکثراً تحت عنوان "TDM wildcard" شناخته می‌شود، به علت اینکه با داشتن چهار پورت RJ11 قادر به فراهم نمودن هر ترکیبی از FXS و FXO می‌باشد؛ یعنی مثلاً می‌توانیم دو تلفن آنالوگ (FXS 2) و دو خط ورودی (FXO 2) به آن وصل نمود. البته واضح است که Asterisk با داشتن ساختار پیمانه‌ای، قادر به افزایش اجزاء جدیدی به سیستم به منظور توسعه سیستم خواهد بود؛ ولی فعلاً همین چهار پورت جوابگوی سیستم آزمایشی ما می‌باشند!



۷-۲. کیفیت خدمات (QoS):

QoS توانایی یک شبکه برای تامین سرویس بهتر برای ترافیک شبکه برگزیده می‌باشد. یکی از بزرگترین چالش‌ها در پیاده‌سازیهای VoIP، به خصوص در کشورهای در حال توسعه؛ اطمینان از داشتن پهنای باند کافی برای تماس‌های تلفنی بدون توجه به میزان اشغالی و ترافیک جاری اتصال اینترنتی می‌باشد. هنگام طراحی یک شبکه VoIP شما باید تلاش کنید که بهینه‌ترین استفاده را از پهنای باند موجود داشته باشید و اختلال سیگنالینگ (Jitter) و تأخیر (Latency) را به حداقل برسانید.

۱۲- از مهمترین و سازگارترین آن‌ها می‌توان به کارت‌های سخت‌افزاری Digium اشاره کرد که در ادامه یکی از آن‌ها بررسی شده و مورد استفاده قرار می‌گیرد.

می‌تواند آن‌ها را به PBX ما متصل نماید، خلاصه شده است. جهت پوشش دادن جامع مطالب، سعی شده تا جایی که ممکن است از تکنولوژی‌های متفاوتی برای هرکدام استفاده شود. در یک سیستم واقعی، شما باید سعی نمایید تا تفاوت تکنولوژی‌ها و تجهیزات بکار گرفته شده را به حداقل ممکن برسانید.

| Extension | تکنولوژی مورد استفاده | سازمان مورد نظر |
|-----------|---|-----------------|
| 462 | تلفن VoIP با بکارگیری پروتکل SIP (تلفن اینترنتی مبتنی بر SIP) | کتابخانه عمومی |
| 463 | تلفن معمولی با واسط ATA مبتنی بر پروتکل SIP | بیمارستان محلی |
| 464 | تلفن معمولی با واسط ATA مبتنی بر پروتکل IAX2 | مدرسه اصلی |
| 466-465 | دو تلفن نرم افزاری با هر دو پروتکل SIP و IAX2 | انجمن روستایی |

<http://www.asterisk.org>

- برای پیکربندی بدنه اصلی، نیاز به Package های "add ons" و "music" ندارید.
- سورس Asterisk نیاز به کامپوننت های دیگری نیز روی سیستم شما دارد، مطمئن شوید Package های زیر را روی سیستم خود نصب نموده‌اید:

- Bison (یک مولد پارسر)
- zlib and zlibdev (جهت فشرده‌سازی - توسعه)
- openssl and openssldev (SSL-کتابخانه‌های توسعه)
- libc6dev (کتابخانه توسعه و هدرهای C جهت GNU)
- gcc and make (کامپایلر C ابزار make)
- کامپایل کردن Asterisk با سایر نرم‌افزارهای لینوکس تفاوت چندانی ندارد:

- برای کامپایل `#make`
- برای نصب `#make install`
- برای نصب و اسکریپت های راه اندازی `#make config`
- برای نصب فایل‌های پیکربندی پیش فرض (مثال‌ها) `make # samples`

به علت این که ما در مراحل آتی نیاز به بکارگیری رابط Digium Wildcard-tm خواهیم داشت؛ باید مازول کرنل درایور Zaptel نیز کامپایل و نصب گردد:

- سورس کد Zaptel را نیز از <http://www.asterisk.org> دانلود نمایید.

متأسفانه مازول درایور Zaptel جزو کرنل لینوکس نبوده و شما بالخصوص، نیاز به کامپایل آن خواهید داشت. از نصب بودن Package های هدر کرنل مطمئن شوید^{۱۳}.

۸-۲. پیش نیازها

نکاتی در زمینه نحوه نصب، پیکربندی و دستورات ابتدایی کار با Asterisk/Trixbox، فایل‌های پیکربندی و انواع کاربران ورودی/خروجی Asterisk مطرح است که در ذیل به آن‌ها به صورت خلاصه اشاره می‌شود:

۸-۲-۱. نصب Asterisk

این موضوع که برای یک مساله راه حل‌های متعددی وجود دارد، کاملاً از دید نویسنده معقول بوده و بنابراین مطمئن باشید که این نرم افزار (Asterisk) با تمام عظمت و اعتبارش، تنها یا بهترین راه حل برای یک IP-PBX نیست؛ بنابراین در به کارگیری راه حل خودتان مردد نباشید و این قسمت را تمرینی برای شروع بدانید.

۸-۲-۱-۱. دانلود/کامپایل نمودن Asterisk

همچون اکثر نرم افزارهای مجانی و سورس باز، دو راه حل برای کار کردن با Asterisk روی سیستم شما وجود دارد. راه اول دانلود Package های pre-compile شده می‌باشد؛ که در این صورت فایل ISO مربوط به AsteriskNow™ را از آدرس:

<http://www.asterisk.org> دانلود کرده و پس از تهیه CD bootable آن، مراحل ساده نصب آن را ادامه دهید (برای ساده شدن کار، در صورت عدم اطلاع از مراحل نصب تمام تنظیمات پیش فرض را تایید کنید)، در نهایت با دادن تنظیمات شبکه؛ IP اتصال به IP-PBX به شما داده می‌شود که با واسط وب یا به صورت دستی از کنسول لینوکس قادر به ادامه مراحل خواهید بود.

راه حل دوم دانلود و کامپایل نمودن Asterisk از سورس کد می‌باشد است؛ اگر شما تصمیم به دانلود Asterisk از سورس گرفته‌اید، نکات زیر کار شما را تسهیل می‌کنند:

- سورس Asterisk را از مسیر روبرو دانلود نمایید:

۱۳- با دستور `uname -a` می‌توانید از نسخه کرنل سیستم خود مطلع شوید.

۸-۲-۱-۲. دستورات اصلی Asterisk

Asterisk دو کامپوننت built-in (توکار) دارد. یک سرور که معمولاً پروسس پشت صحنه است و یک کلاینت که سرور را مانیتور می کند. هر دوی آنها دستور مشابهی "Asterisk" استفاده می کنند ولی با flag های متفاوت.

- راه اندازی/متوقف نمودن Asterisk از مرحله run

```
#/etc/init.d/asterisk (start|stop)
```

- راه اندازی Asterisk از خط فرمان : متناوباً می توانید Asterisk را از خط فرمان اجرا کنید (بعنوان daemon)

```
# asterisk
```

- با پارامتر Asterisk(-VVV) با ذکر جزئیات نمایش داده خواهد و جهت باز شدن یک کنسول کلاینت با پارامتر (-C) شما قادر خواهید بود هر آنچه در سرور Asterisk می گذرد را مانیتور نمایید :

```
# asterisk -vvvc
```

- با پارامتر (-r) اگر سرور Asterisk در حال اجرا باشد، یک ترمینال کلاینت باز کرده و با اتصال به سرور می توانید وضعیت آنرا مانیتور نمایید :

```
# asterisk -r
```

دستورات پایه CLI (کلاینت)

فایل های پیکربندی را مجدداً بارگذاری می نماید.

```
#CLI>reload
```

نمایش اطلاعات debug مربوط به SIP یا IAX2 را فعال می کند.

```
#CLI> IAX2 debug
#CLI> SIP debug
```

نمایش اطلاعات debug مربوط به SIP یا IAX2 را غیرفعال می کند.

```
#CLI> IAX2 no debug
#CLI> SIP no debug
```

وضعیت فعلی user ها، peer ها و کانال های مرتبط با SIP را نمایش می دهد.

```
#CLI> sip show users
#CLI> sip show peers
#CLI> sip show channels
```

وضعیت فعلی user ها، peer ها و کانال های مرتبط با IAX2 را نمایش می دهد.

```
#CLI> iax2 show peers
#CLI> iax2 show users
#CLI> iax2 show channels
```

۸-۲-۱-۳. فایل های پیکربندی

به علت وابستگی تعداد فایل های پیکربندی که نیاز به ویرایش دارند با تکنولوژی به کار گرفته شده توسعه گر و تنوع زیاد فایل های پیکربندی، در این مقاله فقط به توضیح مختصر پنج فایل اصلی که در سناریو ما کاربرد زیادی دارند بسنده کرده و مابقی را به فرصت های آتی جهت سناریوهای پیشرفته تر واگذار می کنیم.

| نام/مسیر فایل | توضیح مختصر |
|-------------------------------|--|
| /etc/asterisk/extensions.conf | جهت طراحی Dialplan و ارتباط کانال ها با یکدیگر (همیشه اجباری) |
| /etc/asterisk/sip.conf | برای پیکربندی کانال های مبتنی بر SIP (تلفن های اینترنتی و تجهیزات مبتنی بر SIP) |
| /etc/asterisk/iax.conf | برای پیکربندی کانال های مبتنی بر IAX2 |
| /etc/asterisk/zapata.conf | برای پیکربندی سخت افزارهایی که واسط PSTN می باشند. توسط خود Asterisk هنگام بالا آمدن استفاده می شود. |
| /etc/zaptel.conf | پیکربندی low-level برای کارت واسط Zaptel. توسط ابزار پیکربندی ztcfg. قبل از راه اندازی Asterisk مورد استفاده قرار می گیرد. |

۸-۲-۲. Friend و Peer. User

یکی از مفاهیم گیج کننده برای اکثر مبتدیان (حداقل یک مدت برای خودم!) در استفاده از Asterisk، معانی Friend و Peer، User در فایل های پیکربندی sip.conf و iax.conf می باشد. این اصطلاحات، جهت طبقه بندی تماس های ورودی/خروجی می باشند. بدین صورت که User اتصالی است که برای ما شناخته شده است (تماس های ورودی) و Peer یک اتصال خروجی است؛ یعنی User با ما تماس می گیرد و ما هم با Peer اتصال برقرار می کنیم. Friend ها می توانند به هر دو نوع قبل (ورودی/خروجی) شامل شوند.

وقتی ما تماسی از یک user یا friend دریافت می کنیم، باید بدانیم که با آن چه کار کنیم! اصطلاح context معین می نماید که چه قاعده ای (یا مجموعه ای از اعمال) از dialplan^{۱۴} باید بر روی این تماس به خصوص اعمال شود.

در فایل extension.conf دقیقاً مشخص می گردد که هر کاربر (تماس ورودی) طبق قسمت مشخصی از dialplan، باید به کدام کانال ارتباطی متصل گردد.

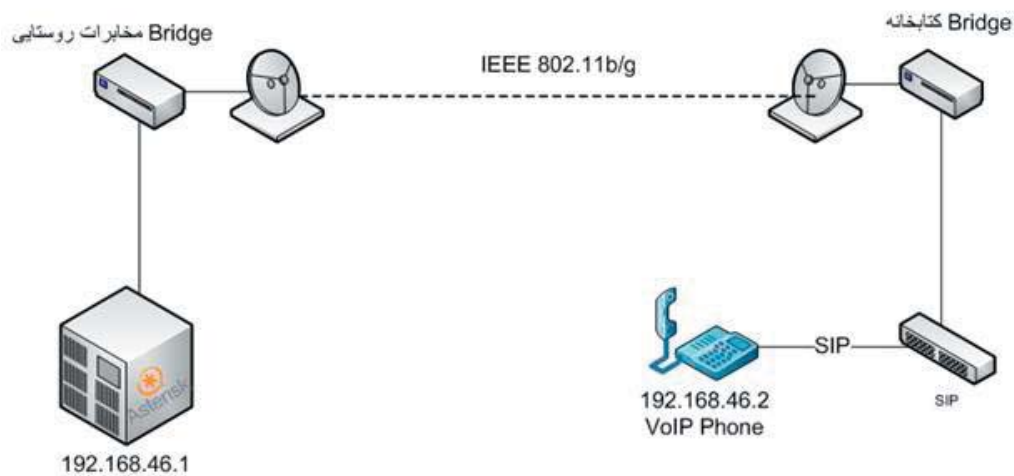
۸-۳. کتابخانه عمومی

اولین کلاینت در کتابخانه عمومی که یک کیلومتر با مخابرات روستایی فاصله دارد قرار می گیرد. تلفن VoIP-SIP مستقیماً توسط اتصال بی سیم (یک اتصال P2P bridged) به IP-PBX ما متصل می گردد. چون IP range تلفن VoIP و PBX یکی هستند، می توانیم نگران مشکلات NAT نباشیم.

| پارامتر مورد نظر | مقدار |
|----------------------------|---------------|
| آدرس IP تلفن VoIP | 192.168.46.2 |
| آدرس (PBX SIP Proxy IP ما) | 192.168.46.1 |
| User name/Auth name | 462 |
| Caller ID | 462 |
| Authentication password | 462pass |
| Codec | G.711 (a-law) |

۱۴- به طور کلی به الگوی عددی شماره تلفن ها در شبکه PSTN اطلاق می گردد که شامل کد کشور، کد محلی و تمام ترکیبات بعدی شماره تلفن می شود، مثلاً ۲ رقم کد کشور، ۳ رقم کد شهر و هفت رقم شماره تلفن و ... ولی در Asterisk یا IP-PBX های مبتنی بر FreePBX؛ به مجموعه ای از context ها گفته می شود که در فایل extension.conf قرار گرفته اند و نحوه تعامل end-point ها و تماسها را معین می نمایند، و قسمت اعظم این فایل پیکربندی را شامل می شوند. رک. به : <http://www.voip-info.org/wiki/index.php?page=Asterisk+Dialplan+Introduction> و یک نمونه آزمایشی از dialplan در Asterisk : <http://www.asteriskblog.com/asterisk-dial-plan-what-is-it/#more-81>

نحوه اتصال تلفن VoIP به IP-PBX در کتابخانه عمومی

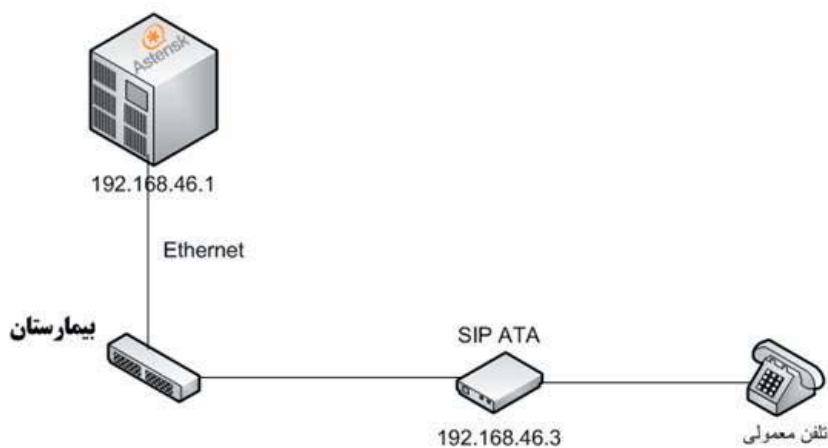


| مقدار | پارامتر مورد نظر |
|---------------|------------------------------|
| 192.168.46.3 | آدرس IP تلفن VoIP |
| 192.168.46.1 | آدرس IP SIP Proxy (PBX) (ما) |
| 463 | User name/Auth name |
| 463 | Caller ID |
| 463pass | Authentication password |
| G.711 (a-law) | Codec |

۴-۸. بیمارستان

دومین کلاینت از شبکه تلفنی داخلی ما یک جعبه ATA است که در بیمارستان محلی تعبیه می شود. فرض می کنیم بیمارستان در فاصله 100 متری مخابرات روستایی قرار گرفته و کابل CAT5 تا آنجا کشیده شده است. پیکربندی ATA هم تفاوت چندانی با تلفن VoIP ندارد و یک صفحه وب برای پیکربندی در اختیار کاربر قرار می دهد.

نحوه اتصال ATA به IP-PBX در بیمارستان



۸-۵. مدرسه اصلی

```
[iaxy_school]
ip: 192.168.46.100
netmask: 255.255.255.0
gateway: 192.168.46.1
codec: alaw
server: 192.168.46.1.2
user: 464
pass: 464pass
register
```

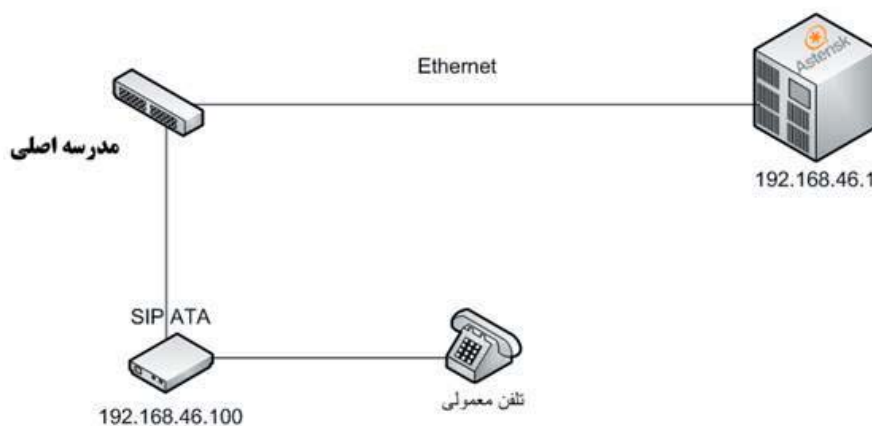
با فرض اختصاص IP 192.168.46.100 به IAXy موردنظر در کنسول Asterisk دستورات زیر را اجرا نمایید:

```
#asterisk -r <ENTER>
#CLI> iax2 provision 192.164.46.100
iaxy_school
```

سومین کلاینت را مدرسه ای همسایه با مرکز مخابرات درنظر می گیریم؛ به علت فاصله کم مدرسه را توسط زوج سیم به مرکز مخابرات روستایی متصل می کنیم. جهت اتصال از یک جعبه ATA دیگر، اما این بار بجای SIP از IAX2 استفاده می نماییم؛ به عنوان نمونه ما از جعبه ATA ساده ای که s101 یا IAXy شناخته می شود و قابلیت اتصال به هرنوع تلفن معمولی آنالوگ را دارد، استفاده می کنیم.

IAXy واسط وب برای پیکربندی ندارد. راحت ترین راه پیکربندی IAXy استفاده از خود Asterisk است. اولین باری که IAXy را وصل می نمایید، به صورت خودکار از DHCP آدرس IP دریافت می کند؛ از سرور DHCP خود IP تخصیص یافته را به دست آورده و در تنظیماتی که در فایل پیکربندی `etc/asterisk/iaxprove.conf` وجود دارد؛ section جدیدی ایجاد نموده و تنظیمات اعمال شده روبرو را به section موردنظر اضافه نمایید(البته بسته به پیکربندی شبکه ممکن است تنظیمات شما مقداری متفاوت باشد):

نحوه اتصال ATA به IP-PBX در مدرسه اصلی



۸-۶. انجمن روستایی

چهارمین کلاینت فرضی ما محل انجمن روستایی می باشد، که فاصله آنرا تا مخابرات روستایی بیست کیلومتر فرض می نماییم؛ برای آشنایی با پروتکل NAT فرض می کنیم در این کلاینت دو کامپیوتر بوسیله NAT بی سیم به مخابرات روستایی متصل هستند. NAT بی سیم دارای IP تخصیص یافته 192.168.46.5 و همچنین کامپیوترها در شبکه داخلی (24/10.10.46.0) می باشند. همانطور که قبلاً ذکر شد یکی از بزرگترین چالش های پیکربندی SIP در یک شبکه مبتنی بر NAT؛ داشتن صدای دوطرفه برای endpoint های SIP می باشد. برای اطمینان از این امر کارهای ذیل را باید درنظر داشته باشید:

۸-۶-۱. در سمت تلفن نرم افزاری:

- فعال نمودن Registration.

• فعال نمودن ^{۱۵} keep-alive بسته‌ها.

• فعال نمودن امکان دریافت صوت روی همان پورتهای که بسته‌های صوتی ارسال می‌شوند.

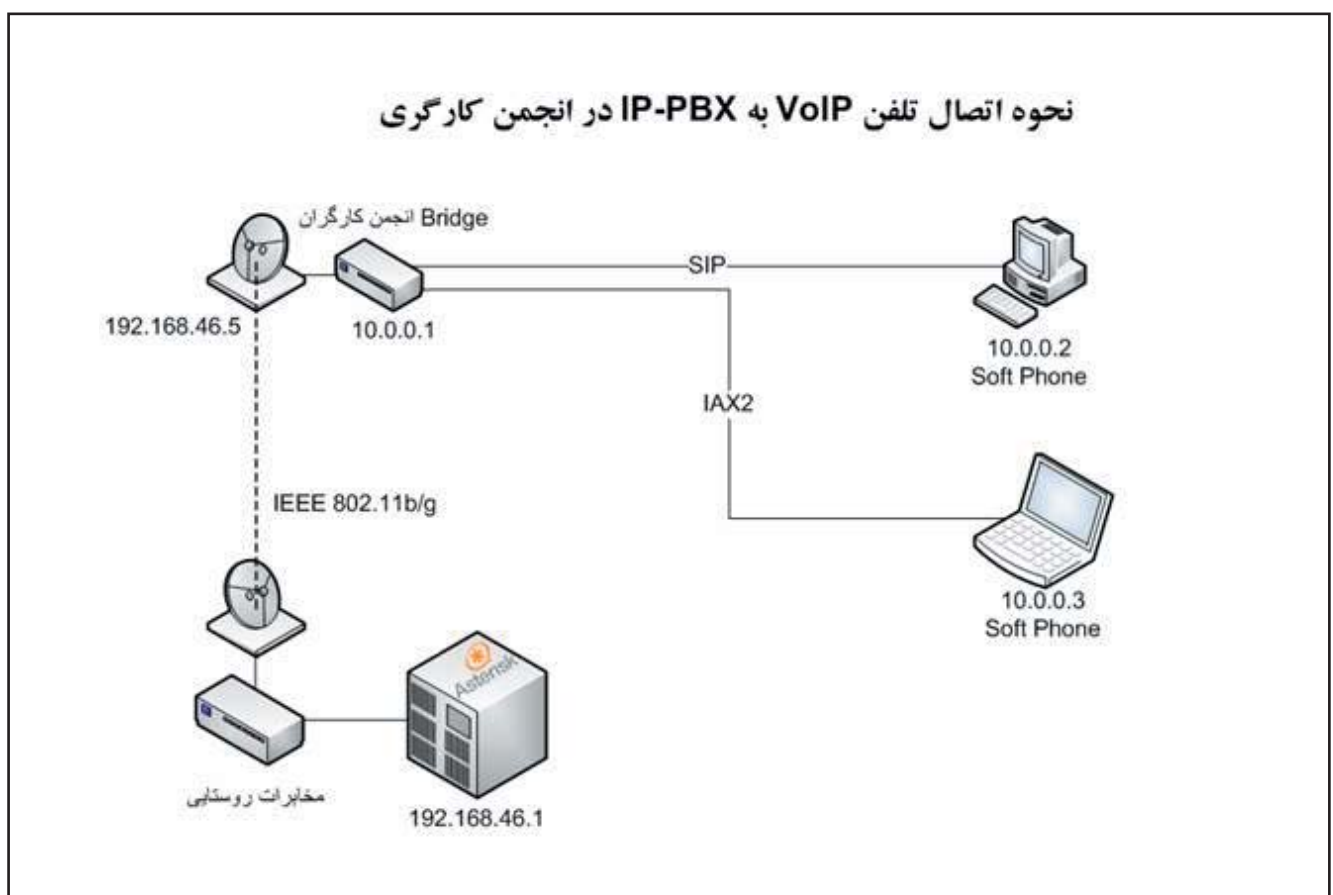
۸-۶-۲. سمت PBX

• اطلاع دادن به Asterisk که تلفن نرم افزاری داخل شبکه NAT قرار دارد.

یک تلفن نرم افزاری مناسب برای SIP. همانطور که قبلاً ذکر شده بود؛ X-Lite^{۱۶} می‌باشد، که بخوبی در شبکه‌های مبتنی بر NAT کار می‌کند.

تلفن‌های نرم افزاری IAX2 داخل شبکه NAT نباید مشکلی داشته باشند، مطمئن شوید که پورت UDP 4569 بسته نباشد. به عنوان نمونه، یک تلفن نرم افزاری مناسب که با IAX2 کار می‌کند ^{۱۷} iaxcomm است.

تنظیمات تلفن نرم افزاری قبلاً به تفصیل توضیح داده شده است، از Username/Password های 465pass/465 و 466pass/466 برای هر کدام از تلفن‌های نرم افزاری استفاده نمایید و مطمئن شوید که codec (G.711 alaw) فعال بوده و IP سرور PBX به 192.168.46.1 تنظیم شده است.



۸-۷. پیکربندی Asterisk

۸-۷-۱. قدم اول: تعریف و پیکربندی کانال‌های ارتباطی

در این سناریو، با توجه به داشتن دو کانال ارتباطی SIP و IAX2 باید دو فایل پیکربندی iax.conf و sip.conf را ویرایش نماییم.

۱۵- بسته‌های Keep-alive، بسته‌هایی خالی "empty" هستند که برای اطمینان از بازبودن NAT برای تماس‌های دریافتی، ارسال می‌گردند.

۱۶- این نرم افزار، در نسخه قبل از این مقاله (مقدمه‌ای بر دنیای VoIP) به تفصیل توضیح داده شده است.

۱۷- می‌توانید این نرم افزار را از لینک روبرو دانلود نمایید: <http://iaxclient.sourceforge.net>

| در فایل sip.conf تغییرات زیر را اعمال نمایید ^{۱۸} | در فایل iax.conf تغییرات زیر را اعمال نمایید |
|---|--|
| <pre>[462] type=friend ; We can call and receive calls secret=462pass context=internal_calls ; All "incoming calls" are associated ; to the context internal_calls host=192.168.46.2 callerid=Library disallow=all ; First we disallow all codecs allow=alaw ; Then we list all the codecs we accept [463] type=friend secret=463pass context=internal_calls host=192.168.46.3 callerid=Hospital disallow=all allow=alaw [465] type=friend secret=465pass context=internal_calls host=dynamic ; We do not know the IP address in advance. ; We expect to learn the IP when the user registers callerid=Farmers1 disallow=all allow=alaw ; NAT specific options follows: nat=yes ; Voice data is sent to IP,port of the NAT qualify=yes ; We send dummy traffic to keep the NAT open</pre> | <pre>[464] type=friend secret=464pass context=internal_calls host=192.168.46.4 callerid=School disallow=all allow=alaw [466] type=friend secret=466pass context=internal_calls host=dynamic ; To learn the visible IP and port ; of the IAX2 client callerid=Farmers2 disallow=all allow=alaw</pre> |

۸-۷-۲. قدم دوم : تعریف قوانین ارتباطی (Rules) و Extension ها (ایجاد Dialplan)

برای کوتاه شدن بحث در این سناریو، ما تمام کاربران را در یک context مشترک به نام [internal_calls] قرار می دهیم^{۱۹}. بنابراین تمامی Dialplan ما در یک context واحد به صورت زیر خلاصه می گردد:

۱۸- توجه نمایید که عبارات بعد از سمی کالن ';' توضیحات می باشند و ضرورتی ندارند.

۱۹- در پروژه های بزرگ و واقعی باید context های متعدد در برگرنده Dialplan سیستم باشند تا از پیچیدگی فزاینده Dialplan جلوگیری گردد.


```
[internal_calls]
exten => 462,1,Dial(SIP/462)
exten => 463,1,Dial(SIP/463)
exten => 465,1,Dial(SIP/465)
exten => 464,1,Dial(IAX2/464)
exten => 466,1,Dial(IAX2/466)
exten => t,1,Hangup() ; Special extension (Timeout)
exten => i,1,Hangup() ; Special extension (Invalid)
exten => s,1,Hangup() ; Special extension (No routing information)
```

FXS را دارا می باشد. در اینجا برای نمونه یک ماژول FXO را در اولین پورت کارت PCI قرار می دهیم.

ساده ترین فایل پیکربندی برای این منظور `etc/zaptel.conf` می باشد؛ Utility پیکربندی Zaptel که به عنوان قسمتی از سورس کد توسعه Asterisk (یا پکیج جداگانه Zaptel) نصب می شود، این فایل را خوانده و مدیریت می کند، برای سخت افزار تهیه شده لازم است حداقل اطلاعات زیر را به این فایل بیفزاییم:

سطر اول به این معنی است که ما از سیگنالینگ FXS از نوع Loopstart روی پورت یک کارت استفاده می کنیم؛ سطرهای دوم و سوم، پارامترهای پیکربندی Zaptel برای نوع tone های به کارگرفته در منطقه^{۲۱} موردنظر را معین می کنند.

```
fxs1s=1
loadzone=ir
defaultzone=ir
```

• نمونه‌ای از خروجی نوعی، می تواند بصورت روبرو باشد:

```
# ztcfg -vv
Zaptel Configuration
=====

Channel map:
Channel 01: FXS Loopstart (Default)
(Slaves: 01)
1 channels configured
```

در این فایل پیکربندی ساده، ما مشخص می کنیم که هر یک از extension های 462 تا 466 با دستور Dial قابل دستیابی بوده و یک کانال SIP یا IAX2 به peer ها با Username مشابه ایجاد می کنند (نسخه نهایی dialplan در بند ۹-۵ آورده شده است، که مرحله به مرحله با تکمیل سناریو، به این قسمت نزدیک تر خواهیم شد).^{۲۰}

۹. اتصال با شبکه PSTN:

در این قسمت سعی خواهیم نمود هریک از کلاینت ها به PSTN دسترسی داشته باشند. در این مقاله تنها به دلیل سادگی استفاده و چندکاره بودن (هر ترکیبی از FXO/FXS)، از کارت PCI شرکت Digium به نام TDM400P استفاده می کنیم.

۹-۱. اضافه نمودن پشتیبانی از کارت TDM400P

• نصب درایور

پس از نصب سخت افزار PCI در شکاف PCI کامپیوتر، باید مطمئن شوید که درایورها کاملاً کامپایل و load شده‌اند. با اجرای `lsmod` شما باید بتوانید درایور load شده `wctdm` را ببینید. همچنین باید کنترلر `zaptel` که آن هم وابسته به نصب بودن `cc_ccitt` می باشد را مشاهده نمایید:

```
# lsmod
zaptel 191748 7 wctdm
crc_ccitt 2304 3 hisax,zaptel,irda
```

• پیکربندی کارت با `ztcfg`

قدم بعدی پیکربندی سخت افزار است. درایورهای `wctdm` نرم‌افزاری همه منظوره برای نسخه های متفاوت کارت TDM۴۰۰P می باشند (به یاد بیاورید که برد PCI قابلیت توسعه چهار FXO/

۲۰- Dialplan های بسیار متفاوت و گسترده ای با ترکیبات دستوری `extension.conf` قابل پیاده سازی می باشند که ذکر جزئیات آن در حوصله این مقاله نمی گنجد، جهت

اطلاعات بیشتر در مورد دستورات `rules` ر.ک. : <http://www.voip-info.org/wiki/view/Asterisk+config+extensions.conf>

۲۱- لیستی از مشخصات tone های موجود را در این لینک ببینید: <http://www.itu.int/ITU-T/inr/forms/files/tones۰۲۰۳.pdf>

```
[channels]
usecallerid=yes
hidecallerid=no
callwaiting=no
threewaycalling=yes
transfer=yes
echocancel=yes
echotraining=yes
context=incoming_pstn
signalling=fxs_ls
channel => 1
```

• پیکربندی Asterisk با بکارگیری سخت افزار Zapata آخرین اقدام ما در این قسمت، پیکربندی Asterisk برای استفاده از سخت افزار است. این کار در فایل `etc/asterisk/zapata.conf` به صورت روبرو انجام می پذیرد؛ سه سطر آخری، مهم ترین عناصر پیکربندی پیش فرض هستند. `context=incoming_pstn` مشخص می کند که تمام تماس های ورودی مربوط به این `context` می شوند. دو خط بعدی سیگنالینگ (`fxs_ls`) و کانال (`channel => 1`) را معین می نمایند. تقریباً از همان موقع که ما این تنظیمات جدید کانال (کانال `zapata`) را به اتمام رساندیم، قادر به دریافت/ارسال مکالمات به PSTN هستیم.

۹-۲. مدیریت تماس های ورودی PSTN

رفتار مطلوب برای یک تماس ورودی PSTN به این صورت است که : وقتی تماسی از طریق خط تلفنی آنالوگ با سیستم برقرار می گردد، تماس گیرنده انتظار یک سیستم پاسخگوی تلفنی (IVR) را دارد که از او در مورد `extension` ای که می خواهد با آن صحبت کند، سوال نماید. با توجه به اینکه ما کلاینت های متعدد VoIP در شبکه تلفنی داخلی خود داریم، باید کاری کنیم که آن ها برای هرکسی که قرار است با خط تلفنی آنالوگ به سیستم ما متصل می شود، قابل دسترس باشند. برای افزودن مقداری هوشمندی به فایل `extensions.conf` section ، زیر را به آن اضافه می کنیم (توضیحات هر خط در صورت نیاز، جلوی آن آمده است ^{۲۲}):

```
[incoming_pstn]
exten => s,1,Answer() ; We answer the call
exten => s,2,DigitTimeout(10) ; Setting Timeout values in seconds
exten => s,3,ResponseTimeout(20)
exten => s,4,Background(vm-extension) ; Voice asking for an extension
exten => i,1,Goto(incoming_pstn,s,1) ; Ask again if invalid extension
exten => t,1,Hangup() ; Hang up if timeout
include => internal_calls ; Makes internal_calls extensions available
```

این دستورات به ترتیب، به این معنی هستند که برای دستیابی به خط PSTN، شما باید در ابتدای شماره عدد صفر را شماره گیری نمایید. دستور `Dial` ارتباطی بین تماس تلفنی با کانال یک `Zaptel` برقرار می نماید، و قسمت دستوری `{EXTEN:1}` / به این معنی است که هنگام شماره گیری (با بیرون)، یک رقم از اول شماره باید برداشته شود.

البته افزودن این `context` به تنهایی کافی نیست؛ ما باید این خط تلفن را به کلاینت هایمان هم معرفی کنیم. ساده ترین راه برای این کار افزودن خط زیر به انتهای section ایجاد شده برای `[internal_calls]` در فایل `extensions.conf` است :

۹-۳. قابل دسترس نمودن PSTN از Dialplan

جهت قابل دسترس شدن PSTN برای تمام کلاینت های VoIP خود، اولاً ما احتیاج به اضافه نمودن `context` جدیدی به فایل `extensions.conf` به صورت روبرو خواهیم داشت:

```
[outgoing_calls]
exten => _0.,1,Dial(Zap/1/
${EXTEN:1})
exten => t,1,Hangup()
```

^{۲۲} - باید توجه داشته باشید که اصوات ضبط شده پیش فرض Asterisk مطمئناً فارسی نخواهند بود! در صورت نیاز شما باید اصوات جدید را با فرمت و `Sample rate` مشابه به

فارسی ضبط، گردآوری و در پوشه مربوطه جایگزین نمایید. برای نمونه می توانید از لیست های گردآوری شده زیر برای اطلاع از متن اصوات استفاده نمایید:

http://www.nathanpralle.com/software/ast_soundlist_extra.html و http://www.nathanpralle.com/software/ast_soundlist.html

```
[channels]
usecallerid=yes
hidecallerid=no
callwaiting=no
threewaycalling=yes
transfer=yes
echocancel=yes
echotraining=yes
```

```
context=incoming_pstn
signalling=fxs_ls
channel => 1
```

```
context=internal_calls
signalling=fxo_ls
channel => 2
```

۵-۹. بروزرسانی Dialplan (ایجاد context های متفاوت)

در dialplan جدید باید موارد زیر را لحاظ کنیم تا مطمئن شویم سیستم آزمایشی ما به صورت عملی قابل استفاده خواهد بود:

۵-۹-۱. اجازه دادن به تماس های ورودی/خروجی به وسیله کانال Zapata

۵-۹-۲. اتصال تلفن آنالوگ متصل به کانال دوم Zapata برای برقراری تماس های ورودی/خروجی با بیرون

با این تفصیل فایل نهایی extensions.conf به صورت زیر درآمده و سناریو ما به پایان می رسد :

```
[incoming_pstn]
exten => s,1,Answer()
exten => s,2,DigitTimeout(10)
exten => s,3,ResponseTimeout(20)
exten => s,4,Background(vm-exten-
sion)
exten => i,1,Goto(incoming_
pstn,s,1)
exten => t,1,Hangup()
include => internal_calls
```

```
include => outgoing_calls
```

۴-۹. اتصال تلفن آنالوگ به PBX

در سناریو پیشنهادی، ما کلاً پنج کلاینت VoIP در چهار محل مجزا مخابرات روستایی قرار دادیم؛ ولی هیچ تلفنی برای آنها فراهم نکردیم. راه آسان برای این کار افزودن یک ماژول FXS به برد TDM400P نصب شده است، که کفایت PBX را خاموش کرده و ماژول جدید را به پورت دوم کارت TDM اضافه نماییم. این ماژول اجازه می دهد که هر نوع تلفن آنالوگی به PBX ما متصل و قابل استفاده گردد.

پس از روشن کردن PBX، با اضافه نمودن خط روبرو به فایل etc/zaptel.conf کارمان با آن تمام می شود :

```
fxs1s=1
fx0ls=2
loadzone=ir
defaultzone=ir
```

برای اطمینان از این که پورت جدید بصورت صحیح شناخته و پیکربندی شده است دستورات ابزار پیکربندی Zaptel باید نتیجه روبرو را به ما بدهند:

```
#ztcfg -vv
Zaptel Configuration
=====

Channel map:
Channel 01: FXS Loopstart (Default)
(Slaves: 01)
Channel 02: FXO Loopstart (Default)
(Slaves: 02)
2 channels configured.
```

از طرف دیگر باید section جدیدی در فایل etc/asterisk/zapata.conf ایجاد نماییم که که تماس هایی که از تلفن آنالوگ می آیند (پورت دوم از کارت TDM که جدیداً آن را نصب کرده ایم) ، را به context مربوط به [internal_calls] مرتبط نماییم.


```
[internal_calls]
exten => 461,1,Dial(Zap/2) ; Extension 461 calls via Zap channel 2
exten => 462,1,Dial(SIP/462)
exten => 463,1,Dial(SIP/463)
exten => 465,1,Dial(SIP/465)
exten => 464,1,Dial(IAX2/464)
exten => 466,1,Dial(IAX2/466)
exten => t,1,Hangup()
exten => s,1,Hangup()
exten => i,1,Hangup()
include => outgoing_calls ; PSTN available to the VoIP clients
[outgoing_calls]
exten => _0.,1,Dial(Zap/1/${EXTEN:1}) ; Remove 0 before dial out
exten => t,1,Hangup()
```

۱۰. نتیجه گیری

این مقاله، تلاشی بود برای این که خواننده را با برخی مباحث ضروری IP تلفنی آشنا نماید. نویسنده با پیگیری یک سناریو ابتدایی، امیدوار است تا توانسته باشد فرصت شروعی در اختیار خواننده جهت راه اندازی یک سیستم VoIP قرار دهد. نکته حائز اهمیت در این زمینه دانستن این مطلب است که پشتکار و تلاش شما تنها کلید موفقیت برای برپا نمودن هر سیستم ترکیبی از شبکه‌های VoIP می‌باشد؛ هیچ مقاله یا کتابی در این زمینه نمی‌تواند همچون تجربه عملی راه اندازی یک سیستم واقعی به شما کمک نماید. در انتها مطمئن باشید که شما در این راه هرگز تنها نخواهید ماند و فروم‌های آنلاین متعددی برای بهره جستن از تجربیات منحصر بفرد شما با دیگر فعالان IP تلفنی و بالعکس، وجود دارند. به جامعه مشتاقان توسعه و پیشرفت VoIP خوش آمدید.

منابع:

- 1- <http://www.voip-info.org>
- 2- <http://www.cisco.com/go/ciscoit>
- 3- <http://www.oreilly.com/catalog/asterisk>
- 4- <http://tools.ietf.org>
- 5- <http://www.it46.se>
- 6- <http://www.astricon.net>

ثبت دامین ✓

از
۱۰,۰۰۰
تومان

ثبت دامین به نام متقاضی
ارایه کنترل پنل تحت وب
توانایی قفل کردن دامین برای امنیت بیشتر (رایگان)
امکان تغییر مشخصات مالک دامین به راحتی
امکان تمدید خودکار دامنه برای جلوگیری از منقضی شدن
امکان تغییر NameServer های دامین توسط کاربر و بدون
نیاز به تماس با ایران هاست



میزبانی وب ✓

از
۴۰,۰۰۰
تومان

ساختار مالتی سرور
ارایه Application Pool اختصاصی
مانیتورینگ خودکار سرویس ها
پشتیبانی مناسب
کنترل پنل مدیریت سرویس ها
تضمین بازگشت وجه تا یک هفته
ارایه گواهینامه SSL



سرورهای اختصاصی ✓

از
۱۱۰,۰۰۰
تومان

ارایه سرورهای پر قدرت Intel و AMD
امکان دسترسی به سرور به صورت Remote
ارایه سیستم عامل های مختلف
ارایه Antivirus و Firewall
ارایه انواع کنترل پنل و نرم افزارهای سرور
ارایه خدمات مدیریت سرور
ارایه سرویس های بکاپ، مانیتورینگ، KVM و بالانسر
قرارگیری در دیتا سنترهای معتبر



سرویس های نمایندگی ✓

از
۲۵۰,۰۰۰
تومان

امکان مشاهده فضای مصرفی کاربران
امکان تخصیص کنترل پنل به هر سایت
امکان استفاده از Name Server های اختصاصی
امکان استفاده از لوگو در کنترل پنل کاربران
امکان اضافه کردن تعرفه های دلخواه در سیستم
امکان محدود کردن فضای هر یک از Domain ها و یا
جلوگیری از استفاده فضای تخصیص داده شده



نویسنده: حمیدرضا متقیان

بررسی تکنیک CSS Sprites و استفاده از آن در ساختن Image Map

m.hamidreza@gmail.com



آیا تا به حال دنبال تکنیکی بوده‌اید تا بتوانید با استفاده از بهینه‌کردن بارگذاری تصاویر بکاررفته در یک صفحه‌ی وب، سرعت بارگذاری را افزایش دهید؟

CSS Sprites تکنیکی ساده و مؤثر است که تأثیر بسزایی در بالا رفتن سرعت بارگذاری صفحات دارد. ابتدا به توضیح این تکنیک می‌پردازیم و در پایان نیز مثال جالبی در مورد نحوه ایجاد یک CSS Image Map با استفاده از CSS Sprites را بررسی خواهیم کرد.

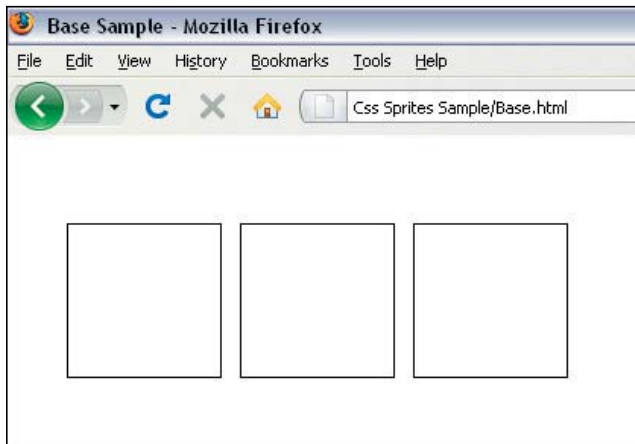
فرض کنید رئیس یک شرکت بزرگ از شما در مورد پایین بودن سرعت بارگذاری یکی از صفحات وب شرکت که باعث ایجاد نارضایتی کاربران شده راه‌حلی می‌خواهد؛ شما به عنوان یک طراح و توسعه‌دهنده‌ی خوب بعد از مطالعه پروسه‌ی بارگذاری صفحه و بررسی درخواست‌ها و پاسخ‌های بین کلاینت و سرور پی به این موضوع می‌برید که بارگذاری حجم بالایی از عکس‌ها در کندشدن این صفحه تأثیر داشته‌است.

نزدیک به ۱۰۰ عکس که سایز آن‌ها بین ۱ تا ۱۰ کیلوبایت است و فقط بالا بودن تعداد آن‌ها باعث بروز این مشکل شده‌است. با این وجود شما چه پاسخی به این سوال خواهید داد؟ آیا خواهید گفت مشکل از سرعت پایین اینترنت کاربران است یا مساله را به بهتر یا بدتر بودن تکنولوژی‌های PHP، JAVA یا ASP.NET نسبت می‌دهید؟

این مقاله به ارائه‌ی راه‌حلی برای این مساله می‌پردازد. تکنیکی به نام CSS Sprites که ایده‌ی اولیه آن از صنعت بازی‌سازی گرفته شده‌است و البته در عرصه وب نیز کاربرد دارد.

ایده اصلی این تکنیک به این صورت است که تمامی عکس‌های کوچک (در اینجا همه ۱۰۰ عکس) در قالب یک تصویر بزرگ قرار خواهد گرفت و با استفاده از CSS مختصات هر عکس کوچک را در تصویر بزرگ پیدا کرده و نمایش می‌دهیم. یکی شدن ۱۰۰ عکس کوچک به یک عکس بزرگ، تأثیر زیادی در پایین آمدن حجم عکس جدید خواهد داشت و سختی کار فقط در تشخیص عکس‌های موردنظر از درون عکس جدید است.

به این مثال توجه کنید:



حال به هر مربع یک عکس نسبت می‌دهیم، کلاس‌های خالی نوشته شده به این صورت اصلاح خواهد شد:

```
.blue
{
    background-image:url('blue.
jpg');
}
.red
{
    background-image:url('red.
jpg');
}
.yellow
{
    background-image:
url('yellow.jpg');
}
```

پس از اجرای صفحه سه مربع به رنگ های آبی، قرمز و زرد خواهید دید:



```
<!DOCTYPE html PUBLIC "-//W3C//DTD
XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/
DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/
xhtml">
<head>
    <style type="text/css">
        .container div
        {
            border: 1px solid;
            float: left;
            height: 100px;
            left: 20px;
            margin-left: 12px;
            margin-top: 50px;
            position: relative;
            width: 100px;
        }
        .blue
        {
        }
        .red
        {
        }
        .yellow
        {
        }
    </style>
</head>
<body>
    <div class="container">
        <div class="blue">
        </div>
        <div class="red">
        </div>
        <div class="yellow">
        </div>
    </div>
</body>
</html>
```

اگر این کد را اجراء کنید ۳ عدد مربع بصورت عکس روبرو خواهید دید:

دادیم و با استفاده از خاصیت `background-position` در سه کلاس `blue`، `red` و `yellow` قسمت مورد نظرمان را از عکس کلی انتخاب کردیم.

استفاده از این تکنیک علاوه بر این که باعث کاهش برآیند حجم عکس‌ها می‌شود به پایین آمدن تعداد درخواست‌ها و پاسخ‌های بین کلاینت و سرور نیز منجر خواهد شد.

پس از اجرای کد بالا همان خروجی قبل را خواهیم داشت با این تفاوت که به جای سه عکس اول، فقط عکس `sprites.jpg` بارگذاری خواهد شد.

در حالت اول حجم صفحه برابر با ۱۲ کیلوبایت و تعداد HTTP Request‌ها برابر با ۴ می‌باشد در حالتی که در حالت دوم، حجم صفحه برابر با ۸ کیلوبایت و تعداد HTTP Request‌ها به ۲ کاهش یافت.

در ادامه برای آشنایی بیشتر با تکنیک `CSS Sprites`، به شرح یک مثال جهت ایجاد `Image Map` خواهیم پرداخت.

مطمئناً تگ‌هایی که برای عکس‌ها در سایت‌هایی مانند `facebook` یا `flickr` می‌توان درست کرد را دیده‌اید. به عنوان مثال در سایت فیس بوک می‌توان اسامی افرادی که در یک عکس قرار دارند را با کادری که مشخص کننده هر فرد است تعریف کرد تا با اشاره موس روی صورت هر فرد، اسم فرد نمایش داده شود.

یکی از روش‌هایی که می‌توان با استفاده از آن این کار را انجام داد `CSS Image Map` می‌باشد که با استفاده از تکنیک `CSS Sprites` قابل انجام است.

کاری که در این مثال می‌خواهیم انجام دهیم اینست که کاربر با قرار دادن موس روی هر یک از شماره‌های موجود در عکس زیر توضیح مربوط به آن شماره در عکس نمایش داده شود.



به عنوان مثال با قرار گرفتن موس بر روی شماره ۴ که یک نوت بوک است این اتفاق بیفتد: به عکس صفحه بعد توجه کنید.

مسأله‌ای که در این حالت وجود دارد این است که سه بار درخواست بارگذاری برای سه عکس مختلف به سرور ارسال شده‌است. دقیقاً همین نمایش را می‌توان با استفاده از تکنیک `CSS Sprites` انجام داد با این تفاوت که در این حالت ما یک عکس را بارگذاری کرده و با استفاده از `CSS` قسمتی از عکس اصلی را که مورد نظرمان هست به کلاس‌های تصویر پیش زمینه انتساب می‌دهیم. با استفاده از نرم افزار فتوشاپ سه عکس را بصورت زیر به یک عکس تبدیل می‌کنیم:



سپس تغییرات ذیل را لحاظ می‌کنیم:

```
.container div
{
    border: 1px solid;
    float: left;
    height: 100px;
    left: 20px;
    margin-left: 12px;
    margin-top: 50px;
    position: relative;
    width: 100px;
    background-image:
    url('sprite.jpg');
}
.blue
{
    background-position: -
    100px 0px;
}
.red
{
    background-position: -
    200px 0px;
}
.yellow
{
    background-position: 0px
    0px;
}
```

در این کد تصویر جدید را که از ترکیب سه تصویر قبلی ایجاد کردیم به عنوان تصویر پیش‌زمینه به کلاس `container` نسبت

عکس نهایی:



CSS Image Maps



در فایلی که ضمیمه شده است نمونه این مثال جهت دانلود قرار دارد. ابتدا کلاس officeMap را بررسی می‌کنیم:

```
dl#officeMap{
    margin: 0;
    padding: 0;
    background: transparent
    url(office.jpg) top left no-repeat;
    height: 262px;
    width: 350px;
    position: relative;
}
```

در حالتی که موس روی هر کدام از شماره‌ها قرار می‌گیرد آیتم موردنظر map می‌شود و همان‌طور که در شکل نمایش داده شده کادری با حاشیه سفید نمایان می‌شود. تصویر این کادرها را با استفاده از فتوشاپ ایجاد می‌کنیم و از آنجایی که در پروژه از تکنیک CSS Sprites استفاده کرده‌ایم عکس‌ها را به هم متصل می‌کنیم. علت وجود عکس سوم در شکل زیر این است که کادر نوت بوک با کادر مانیتور و فلاپی هم‌پوشانی دارد و به این دلیل در یک تصویر مجزا این کادر را به تصویر اضافه کردیم. در نهایت عکس office.jpg که در عکس پیش‌زمینه کلاس officeMap قرار دارد به این صورت درخواهد آمد:

از آنجایی که ۵ شماره در عکس داریم نیاز هست تا ۵ گروه کد CSS برای هر شماره ایجاد کنیم. تنها نکته‌ای که حایز اهمیت است مشخص کردن هر کادر در تصویر است که موقعیت هر عکس را در ویژگی background-image هر کلاس مشخص کرده‌ایم. کدی که برای مانیتور نوشته شده است به این صورت است که آنرا در صفحه بعد مشاهده می‌کنید:

```

dd#monitorDef{ top: 65px; left: 114px; }
dd#monitorDef a{ position: absolute; width: 73px; height: 69px; text-decora-
tion: none; }
dd#monitorDef a span{ display: none; }
dd#monitorDef a:hover{ position: absolute; background: transparent url(office.
jpg) -109px -317px no-repeat; top: -10px; left: -5px; }
dd#monitorDef a:hover span{
    display: block;
    text-indent: 0;
    vertical-align: top;
    color: #000;
    background-color: #F4F4F4;
    font-weight: bold;
    position: absolute;
    border: 1px solid #BCBCBC;
    bottom: 100%;
    margin: 0;
    padding: 5px;
    width: 250%;
}

```

- برای ۴ شماره دیگر نیز کدها به همین صورت است. این مثال در مرورگرهای Mozilla، IE6+، و Opera تست شده است. دانلود کدهای مقاله:

<http://adv.barnamenevis.org/download/files/csssPrites.zip>

منابع:

<http://www.codeproject.com/Articles/35118/Optimize-your-Pages-using-CSS-Sprites.aspx>
<http://www.frankmanno.com/ideas/css-imagemap>

▶ The Myths of Innovation ◀

معرفی کتاب

نویسنده: مهدی عسگری

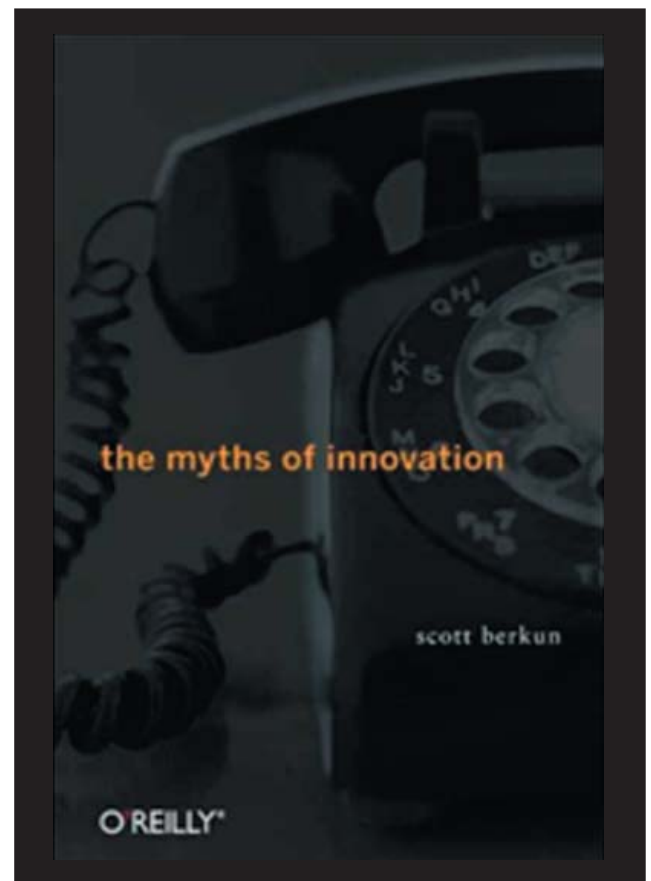
بخشیده و به منظور نوشتن کتاب از شرکت جدا شد. در سال ۲۰۰۵ کتاب معروف The Art of Project Management را نوشت (انتشارات O'Reilly). وی در حال حاضر از طریق نوشتن، سخنرانی و تدریس زندگی می‌گذراند. ایشان در دانشگاه واشنگتن مشغول به تدریس "تفکر خلاقانه" است. وب سایت و وبلاگ شخصی وی در آدرس www.scottberkun.com واقع است. خب، برگردیم به کتاب.

اگر زیاد کتاب‌های زبان اصلی خوانده باشید، احتمالاً با دیدن طرح جلد یک کتاب به ناشر آن پی خواهید برد؛ مثلاً Apress با کتاب‌های زرد و سیاه و کندویی‌اش، Microsoft با کتاب‌های جلد مشکی با طرح ابزار، Wrox با کتاب‌های جلد قرمز با عکس نویسندگان و ... O'Reilly هم اغلب کتاب‌هایش یک عکس (معمولاً عکس یک حیوان) روی جلد دارند و در یک صفحه‌ای در اواخر کتاب با نام Colophon، دربارهٔ حیوان روی جلد توضیح می‌دهند. البته این کتاب عکس حیوان ندارد (عکس یک تلفن است) اما متفاوت‌ترین Colophon را در این کتاب خواهید خواند. یک بخش از آن را در اینجا می‌آورم:

Page numbers were hand-carved, based on a Dutch interpretation of a sketch of reproductions of a famous 13th-century Chinese monograph series believed to have been glanced at once by Marco Polo's best friend's sister. Note the glory of the sans-serifed ascenders! They cost extra, you know.

The ink that makes up these very words was extracted from thousands of adolescent Malaysian juniper beetles, hand-picked for their deep black hues

در این شماره کتابی متفاوت برایتان در نظر گرفته‌ام؛ این کتاب مستقیماً ربطی به برنامه‌نویسی ندارد ولی خواندن آن برای هر مهندسی در هر رشته‌ای از واجبات است!



کتاب The Myths of Innovation همانطور که از نامش پیداست سعی در معرفی افسانه‌ها یا دروغ‌های مربوط به خلاقیت دارد و به نوعی خلاقانه، طرز تفکر خواننده را دربارهٔ خلاقیت عوض می‌کند. راستی یادم رفت بگویم که نویسندهٔ این کتاب، آقای Scott Berkun پنج سال در تیم Internet Explorer در مایکروسافت کار کرد و سپس عطای کار در این شرکت را به لقایش

مقاله‌ها و خودآموزهای اینترنتی چیزهای خوبی برای آموزش سریع یک کار در زمان کوتاه و به صورت سطحی هستند (به طوری که در آن لحظه کارمان راه بیفتد) اما هیچ وقت جایگزین کتاب نخواهند شد.

یک کتاب خوب بیش از یک سال از وقت نویسنده‌اش را می‌گیرد، توسط چندین نفر افراد خبره در آن زمینه بررسی می‌شوند تا از نظر فنی صحیح باشد، توسط ویراستاران ادبی ویرایش می‌شود و در کل از فیلتر چندین نفر می‌گذرد تا چاپ شود، اما مقاله‌ها (عموماً) اینطور نیستند؛ مقاله فقط سعی در آشنایی با یک چیز (زبان، مفهوم، برنامه، ...) دارد اما در یک کتاب شما مسیری را طی می‌کنید و در طی این مسیر با خواندن فصل‌ها، مطالعه مثال‌ها و وقت گذاشتن بر روی تمرین‌ها به مطلب تسلط (نسبی) پیدا می‌کنید. البته باید یادآور شوم که خواندن کتاب بدون انجام کار عملی (لااقل در رشته‌ی عملی‌ای مثل نرم‌افزار و برنامه‌نویسی) چندان حاصلی در بر نخواهد داشت و به مرور به دست فراموشی سپرده می‌شود. امیدوارم همیشه کتاب خوان باشید



کتاب حدود ۱۹۰ صفحه است و سال ۲۰۰۷ توسط O'Reilly منتشر شده و متشکل از ۱۰ فصل و یک ضمیمه است.

فهرست فصل‌ها به ترتیب:

- 1- The myth of epiphany
- 2- We understand the history of innovation
- 3- There is a method for innovation
- 4- People love new ideas
- 5- The lone inventor
- 6- Good ideas are hard to find
- 7- Your boss knows more about innovation than you
- 8- The best ideas win
- 9- Problems and solutions
- 10- Innovation is always good

در هر فصل دلایل متعدد مؤلف همراه با ذکر مثال‌های مختلف از تاریخ را خواهید خواند که عنوان هر فصل را رد می‌کنند! یعنی همین الان نیز با خواندن تیتیر هر فصل تقریباً می‌دانید در آن فصل به چه چیزی پرداخته خواهد شد.

تمامی فصول نیز حاوی عکس هستند. نکته متفاوت دیگر در مورد این کتاب شیوه ارجاع به پاورقی‌ها است که به جای عدد از علامت‌های مخصوصی استفاده کرده است.

مثال‌های کتاب لزوماً قدیمی و کلیشه‌ای نیستند، بلکه از نرم‌افزارها و شرکت‌های نرم‌افزاری، موبایل و اتوموبیل هم استفاده شده است. در کل خواندن این کتاب (مثل دیگر کتاب‌های خوبی که به دقت گلچین کرده و در اینجا معرفی می‌کنم و شرط معرفی‌اش این است که خودم خوانده باشم) لذت بخش است.

عادت خوبی (؟) که دارم این است که همیشه بین هر دو سه کتاب فنی‌ای که می‌خوانم، یک کتاب عمومی یا غیرفنی هم بینشان مطالعه می‌کنم تا افق دیدم نسبت به حرفه‌ام وسیع‌تر شود؛ سعی‌ام در این سری ستون‌ها این است که کتاب‌هایی معرفی کنم که گذر زمان غبار کهنگی روی آن‌ها نمی‌پاشد (فکر کنم خیلی ادبی شد!) نه کتاب‌هایی که مختص یک زبان یا سیستم عامل یا فن‌آوری باشند که دو سه سال بعد کهنه شوند. این کتاب جزو دسته اول است.

بعضی‌ها فکر می‌کنند در عصر کنونی و با این سرعت پیشرفت، دیگر جایی برای خواندن کتاب نمانده و دیگر مقاله‌های آنلاین و ویکی‌ها و ... جای کتاب را خواهند گرفت که باید بگویم صد در صد مخالف این طرز تفکر هستیم.

ISAPI Extension چیست؟

mehdi_mousavi@hotmail.com



Interface یا همان ISAPI، مجموعه رابط‌های برنامه‌نویسی است که روشی قدرتمند برای توسعه وظایف IIS در اختیار توسعه‌دهندگان نرم افزار قرار می‌دهد. اگرچه، ISAPI Extension ها به هیچ وجه محدود به IIS نمی‌شوند، اما عموماً از آن‌ها در ارتباط تنگاتنگ با IIS استفاده می‌شود.

مقایسه CGI و ISAPI

توسعه یک برنامه CGI عبارت است از ایجاد فایل EXE ای توسط زبان‌های برنامه‌نویسی C، C++ یا Perl. این فایل EXE به ازای هر درخواست اجرا شده و سپس خاتمه می‌یابد که این امر باعث استفاده مفرط از حافظه هنگامی که کاربران یک صفحه را بارها درخواست می‌کنند، می‌گردد.

این استفاده مفرط از حافظه که می‌تواند باعث از کار افتادن سرور به طور کامل شود، در ISAPI Extension ها حل شده است. ISAPI Extension یک DLL عادی است که ۳ تابع ویژه دارد. این توابع توسط پروسه فراخوان (بعبارت دیگر، IIS) صدا زده می‌شوند و فارغ از تعداد Client هایی که قصد استفاده از آن را به صورت همزمان دارند، تنها یک بار در حافظه بارگذاری می‌شوند. (ایده خوبی است اگر بتوانید به مرجعی رجوع کنید تا متوجه شوید که مدیریت حافظه تحت Windows 2000 چگونه انجام می‌شود. کتاب **Visual C++ 6.0 Bible**، فصل ۱۸ - مدیریت حافظه، این کار را به خوبی انجام داده است).

مفاهیم اساسی ISAPI

از آنجایی که ISAPI Extension و پروسه فراخوان (IIS) در یک فضای آدرس دهی یکسان قرار دارند، می‌توانند با یکدیگر مستقیماً ارتباط برقرار کنند. این به معنای داشتن قوه پتانسیل بسیار بزرگی برای از کار انداختن IIS و در برخی مواقع، کل وب سرور است. به شکل زیر بنگرید:

Extension چیست؟

این مقاله، ISAPI Extension ها را به تفصیل بررسی کرده و به شما نشان خواهد داد که چگونه می‌توانید Extension ای به منظور اعتبار سنجی شماره کارت اعتباری، طراحی کنید.

مقدمه

احتمالاً سایت‌های بسیاری را دیده‌اید که پس از مرور، آدرس این سایت‌ها به فایلی با پسوند DLL در شاخه اسناد آن Domain ختم می‌شود، مگر آنکه غارنشین بوده باشید. چیزی شبیه این URL مجازی:

<http://www.mydomain.com/script/example.dll?ID=p05874&Tx=870250AZT6>

هدف این DLL چیست و چه ارتباطی با مقاله امروز دارد؟

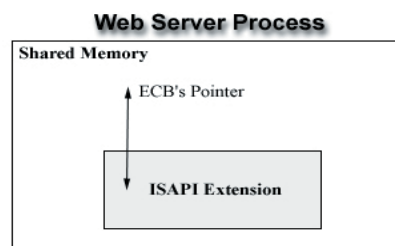
این گونه DLL ها توسط Internet Server API یا به طور مختصر ISAPI ایجاد می‌شوند. ISAPI برای غلبه بر کاستی‌های Common Gateway Interface یا CGI توسعه یافت. اگرچه امروزه سایت‌های جدیدی که صرفاً توسط CGI طراحی شده‌اند، را تجربه می‌کنیم، با این حال، ISAPI DLL ها توانایی‌هایی دارند که CGI به هیچ طریق ممکن نمی‌تواند آن‌ها را به ما ارائه کند.

من قصد دارم تا این مقاله را، با جزییات اساسی که هر برنامه‌نویس ISAPI به منظور طراحی بهتر یک ISAPI Extension باید از آن‌ها مطلع باشد، آغاز کنم. سپس، جزییات طراحی یک ISAPI Extension سودمند را گام به گام توضیح خواهم داد. این Extension قادر است تا شماره کارت اعتباری مورد نظرتان را اعتبار سنجی کند. بله! این مقاله همچنین پاسخ من به افرادی است که در مورد الگوریتم اعتبار سنجی کارت‌های اعتباری، پی در پی سوال کردند.

ISAPI چیست؟

Internet Server Application Programming

همانطور که می‌بینید، Extension شما با هر مشکلی مواجه شود، می‌تواند در صورت عدم کنترل مناسب، روی کل پروسه Web Server تاثیر گذارد. همانطور که در شکل فوق نمایش داده شده است، ارتباط بین Extension و IIS از طریق اشاره‌گری به ساختار ECB یا Extension Control Block که به صورت زیر تعریف شده، میسر است:



```
typedef struct _EXTENSION_CONTROL_BLOCK
{
    DWORD      cbSize;                // size of this struct.
    DWORD      dwVersion;             // version info of this spec
    HCONN      ConnID;                // Context number not to be modified!
    DWORD      dwHttpStatusCode;      // HTTP Status code
    CHAR       lpszLogData[HSE_LOG_BUFFER_LEN]; // null terminated log info

    LPSTR      lpszMethod;            // REQUEST_METHOD
    LPSTR      lpszQueryString;       // QUERY_STRING
    LPSTR      lpszPathInfo;          // PATH_INFO
    LPSTR      lpszPathTranslated;    // PATH_TRANSLATED

    DWORD      cbTotalBytes;          // Total bytes indicated from client
    DWORD      cbAvailable;           // Available number of bytes
    LPBYTE     lpbData;               // pointer to cbAvailable bytes

    LPSTR      lpszContentType;       // Content type of client data

    BOOL (WINAPI * GetServerVariable) (HCONN hConn,
    LPSTR      lpszVariableName,
    LPVOID     lpvBuffer,
    LPDWORD    lpdwSize );

    BOOL (WINAPI * WriteClient) (HCONN ConnID,
    LPVOID     Buffer,
    LPDWORD    lpdwBytes,
    DWORD      dwReserved );

    BOOL (WINAPI * ReadClient) (HCONN ConnID,
    LPVOID     lpvBuffer,
    LPDWORD    lpdwSize );

    BOOL (WINAPI * ServerSupportFunction) ( HCONN hConn,
    DWORD      dwHSERequest,
    LPVOID     lpvBuffer,
    LPDWORD    lpdwSize,
    LPDWORD    lpdwDataType );

} EXTENSION_CONTROL_BLOCK, *LPEXTENSION_CONTROL_BLOCK;
```

اگر extension شما حاوی این تابع باشد، هنگام شروع اولیه و خاتمه extension شما، فراخوانی می شود. وضعیت توسط پارامتر dwCallReason تعیین می گردد که می تواند یکی از مقادیر از پیش تعریف شده ذیل باشد:

- DLL_PROCESS_ATTACHED
- DLL_THREAD_ATTACH
- DLL_THREAD_DETACH
- DLL_PROCESS_DETACH

توضیح هر یک از این پارامترها و جزئیات هر یک، از حوصله این مقاله خارج است. لطفاً برای اطلاعات بیشتر به سایت MSDN مراجعه نمایید.

در هر حال، می توانیم پارامتر hModule را برای استفاده های بعدی در ماژول خود نگهداری کنیم (اگر مطلوب ما باشد) و به سادگی مقدار TRUE را بازگردانیم. هنگامی که در حال توسعه یک extension هستیم، عموماً کاری برای انجام در این تابع نداریم.

GetExtensionVersion، نقطه ورودی واقعی

این تابع در حقیقت اولین تابعی است که توسط IIS از ماژول ما فراخوانی می شود تا اطلاعاتی در مورد extension ما کسب کند. برای فهمیدن بهتر این مطلب، اجازه دهید تا به الگوی این تابع نگاهی کنیم:

```
BOOL WINAPI GetExtensionVersion(HSE_
VERSION_INFO *pVer);
```

به محض فعال شدن این تابع، موظف هستیم تا اطلاعات extension خود را در پارامتر pVer پاس شده به تابع، قرار دهیم. نوع داده ای این اشاره گر، HSE_VERSION_INFO می باشد که به صورت زیر تعریف شده است:

```
typedef struct _HSE_VERSION_INFO
{
    DWORD dwExtensionVersion;
    CHAR lpszExtensionDesc[HSE_MAX_
EXT_DLL_NAME_LEN];
} HSE_VERSION_INFO, *LPHSE_VERSION_
INFO;
```

که در آن dwExtensionVersion نسخه extension و lp szExtensionDescription توضیحاتی در مورد extension است. اگر از درون این تابع مقدار TRUE را بازگردانیم، به IIS گفته ایم که آماده ایم extension ما مورد بهره برداری قرار گیرد. در غیر این صورت، IIS از extension ما استفاده نخواهد کرد.

هرگونه ارسال اطلاعات بین پروسه فراخون و Extension، از طریق این بلوک کنترلی صورت می گیرد. ما به زودی نگاهی به ساختار ECB خواهیم انداخت. در حال حاضر، اجازه دهید تا ببینیم چگونه در ارتباط تنگاتنگ با Extension شما کار خواهد کرد تا پاسخ بیننده سایت شما را ارسال کند.

هر گاه که یک extension، مورد دستیابی قرار می گیرد (بعنوان نمونه، <http://www.mydomain.com/script/example.dll?D=p05874&Tx=870250AZT6>)، IIS

بررسی می کند که آیا example.dll در حافظه بار شده است، یا خیر. در صورتی که پاسخ منفی باشد، بارگذاری DLL مزبور آغاز می شود. هنگامی که DLL در حافظه قرار گرفت، worker thread ای برای مدیریت extension ما فعالیتش را آغاز می کند و سپس، مدخل ورودی برنامه (تابع DLLMain) فراخوانی می شود. هنگامی که اجرای این تابع به پایان رسید، سرور تابع GetExtensionVersion را به منظور انجام دو عمل، فراخوانی می کند:

۱. رد و بدل کردن اطلاعات مربوط به نسخه

۲. گرفتن رشته ای حاوی توضیح مختصری درباره extension سپس، سرور تابع HttpExtensionProc را با پاس کردن یک کپی از اشاره گر ECB به آن، برای شروع عملیات اصلی extension، فراخوانی می کند. این تابعی است که امکان ارسال داده ها به client را ممکن می سازد. ما این تابع را به زودی بررسی خواهیم کرد.

سومین و آخرین نقطه ورودی در یک ISAPI Extension DLL، تابع TerminateExtension است که هنگام تخلیه بار extension از حافظه، فراخوانی می شود. تمامی کدهای مربوط به پاکسازی در این تابع قرار می گیرند. به طور خلاصه، ISAPI Extension یک DLL عادی است که سه تابع را برای تعامل با سرور پیاده سازی می کند:

۱. GetExtensionVersion

۲. HttpExtensionProc

۳. TerminateExtension (اختیاری)

با در دست داشتن این اطلاعات، اجازه دهید تا از DLLMain آغاز کنیم، نقطه شروع هر DLL ای.

DLLMain، نقطه ورودی

همانطور که مایکروسافت می گوید، "تابع DLLMain نقطه ورودی دلخواه در یک DLL است. در صورت استفاده، این تابع توسط سیستم هنگام شروع اولیه یا خاتمه پروسه ها و Thread ها، یا به هنگام فراخوانی توابع LoadLibrary و FreeLibrary، فراخوانی می شود." الگوی این تابع در ذیل آمده است:

```
BOOL APIENTRY DllMain(HANDLE hModule,
    DWORD dwCallReason, LPVOID lpReserved
```


اهداف

ما قصد داریم تا یک ISAPI-Extension غیر MFC، به کمک توابع Win32 تولید کنیم تا شماره هر کارت اعتباری (Master Card) مورد نظری را، اعتبار سنجی کند. ما نام این extension را validate.dll می گذاریم و در آن، به سادگی برای client پیامی ارسال می کنیم تا بداند شماره کارت اعتباری مورد نظرش معتبر است، یا خیر.

البته می بینیم که چه اتفاقی خواهد افتاد اگر کاربر بخواهد صفحه را به این روش:

http://mydomain/script/validate.

dll?some%20string

یا

http://mydomain/script/validate.dll?

و یا URL های غیر معتبر دیگر (البته از نگاه extension ما، زیرا URL های مزبور از دید مرورگر معتبر هستند) مشاهده کند. و این تمام کاری است که extension ما انجام می دهد.

چرا MFC نه؟

اگر چه MFC عمل تجزیه و تحلیل Query String ها را به طرز چشمگیری ساده می کند، اما به طرز قابل توجهی باعث افزایش حجم extension ما نیز خواهد شد. از طرف دیگر، اگر شما زیر و بم چگونگی کارکرد یه extension غیر MFC را بدانید، یک گام در ساخت extension های MFC جلوتر هستید. بدین دلیل در اولین ISAPI Extension خودمان از MFC اجتناب می کنم.

الگوریتم Luhn

اکنون سوال این است که چگونه می توانیم اعتبار یک "کارت اعتباری" را بسنجیم. روش مورد استفاده، الگوریتم Luhn نام دارد (تا آنجاییکه اطلاع دارم، به این الگوریتم الگوریتم Sum10 نیز گفته می شود). این شماره را به عنوان نمونه در نظر بگیرید:

۵۱۶۸۲۵۴۲۳۶۰۲۱۵۴۸

برای آن که بدانیم این شماره یک کارت اعتباری معتبر است، باید ۴ مورد ساده زیر را طی کنید:

۱. با شروع از چپ ترین رقم، رقم ها را یکی در میان در ۲ ضرب می کنیم. به عبارت دیگر، ما ارقام Bold را در دو ضرب می کنیم:

5168254236021548

5 * 2 = 10

6 * 2 = 12

2 * 2 = 4

4 * 2 = 8

3 * 2 = 6

0 * 2 = 0

1 * 2 = 2

4 * 2 = 8

HttpExtensionProc، نقطه ورودی اصلی

بخش سرگرم کننده هر ISAPI Extension ای هنگامی آغاز می شود که تابع HttpExtensionProc فراخوانی می شود. همانطور که احتمالاً به خاطر می آورید، این رویه ای است که ارسال داده ها به Client را میسر می سازد. برای مشاهده چگونگی انجام این کار، اجازه دهید تا نگاهی به الگوی این تابع داشته باشیم:

```
DWORD WINAPI HttpExtensionProc(EXTENSION_CONTROL_BLOCK *pECB);
```

که در آن، pECB، اشاره گر به بلوک کنترلی extension است که امکان مرادده سرور و extension را ممکن می سازد. این شما هستی که در این تابع، تصمیم می گیرید صفحه وب باید حاوی چه چیزی باشد و چگونه آن را به کاربر نمایش دهید. اما چگونه؟ آیا اعضای ساختار ECB را بخاطر می آورید؟ ECB حاوی متدی با الگوی زیر است:

```
BOOL WriteClient(HCONN ConnID, LPVOID Buffer, LPDWORD lpdwBytes, DWORD dwSync);
```

با استفاده از این تابع، می توانید اطلاعات موجود در Buffer را به سمت Client ای که با شناسه ConnID تعیین شده است، ارسال کنید، Client ای که درخواست کرده است. به عنوان مثال، برای ارسال **یک رز قرمز بزرگ**، می توانید بدین گونه عمل کنید:

```
char szBigRedRos[] = "<font color='#FF0000' size='3'><b>رز قرمز</b></font>";
DWORD dwSize = strlen(szBigRedRos);
pECB->WriteClient(pECB->ConnID, szBigRedRose, dwSize, 0);
```

جالب است، نه؟ گمان می کنم که دیگر حداقل اطلاعات مورد نیاز برای توسعه اولین ISAPI Extension خود را داشته باشید. پس شروع می کنیم.

نیازمندی های پروژه

- اندکی صبر و تحمل
- کامپایلر MS-VC++ 6.0
- ویندوز 2000 نسخه Advanced Server به همراه IIS نصب شده
- یک مرورگر

```

if(strlen(pszNumber) != 16)
    return ERR_WRONG_NUMBER_OF_DIGITS;

for(int i = 0; i < 16; i++)
    if(!isdigit(pszNumber[i]))
        return ERR_INVALID_INPUT;

if(pszNumber[0] != '5' ||
pszNumber[1] < '1' || pszNumber[1]
> '5')
    return ERR_NOT_A_MASTERCARD;

int nSum;
for(i = 0, nSum = 0; i < 16; i +=
2)
{
    int nDigit = (pszNumber[i] -
48) * 2;
    nSum += (nDigit < 10 ? nDigit :
nDigit / 10 + nDigit % 10)
        + (pszNumber[i + 1]
- 48);
}

if(nSum % 10)
    return ERR_INVALID_CC;

return 0;
}

```

و در نتیجه، برای آزمودن شماره یک Master Card، کافیت تا تابع را بدین صورت فراخوانی کنید:

```

BYTE byRet = CheckCC("1269875230210
254");
if(!byRet)
{
    //this is a valid master card #
}
else
{
    //An invalid master card#, byRet
shows the error code!
}

```

۲. ارقام حاصله را به صورت تکی با هم جمع کنید:

$$1 + 0 + 1 + 2 + 4 + 8 + 6 + 0 + 2 + 8 = 32$$

۳. ارقام دست نخورده را به حاصل بیفزایید:

$$1 + 8 + 5 + 2 + 6 + 2 + 5 + 8 = 37$$

۴. نتیجه گام های ۲ و ۳ را به یکدیگر افزوده، نتیجه را بر ۱۰ تقسیم کنید:

$$32 + 37 = 69$$

$$69 \% 10 = 9$$

۵. اگر حاصل صفر است، این یک کارت معتبر است. در غیر این صورت، شماره کارت غیر معتبر است.

بنابراین ۵۱۶۸۲۵۴۲۳۶۰۲۱۵۴۸ شماره غیر معتبری است، زیرا باقیمانده ۶۹ بر ۱۰ عدد ۹ است، نه صفر.

آزمون های بیشتر

با تکمیل الگوریتم Luhn، می توانیم یک گام به پیش برویم و نوع کارت اعتباری را بر اساس جدول زیر تشخیص دهیم:

| Credit Card | Prefix | Length (digits) |
|------------------|--------|-----------------|
| Master Card | 51-55 | 16 |
| VISA | 4 | 13, 16 |
| American Express | 34, 37 | 15 |

اما برای این مقاله، ما فرض می کنیم که کارت اعتباری مورد نظر Master Card است، بنابراین شماره را بر اساس این فرض می آزمایشیم؛ به بیان دیگر، ما تنها از ۱۶ کاراکتر بودن کارت اطمینان حاصل می کنیم. شما می توانید در صورت نیاز، این تابع را توسعه دهید.

چگونه الگوریتم Luhn را در C پیاده سازی کنیم؟

در ذیل، تابعی را که نوشته ام می بینید که بر اساس رشته ورودی (شماره کارت اعتباری)، مقدار صفر را در صورت معتبر بودن شماره باز می گرداند. در غیر این صورت، عددی غیر صفر بر می گرداند که بیانگر کد خطا است:

```

#define ERR_WRONG_NUMBER_OF_DIGITS
1
#define ERR_NOT_A_MASTERCARD
2
#define ERR_INVALID_CC
3
#define ERR_INVALID_INPUT
4

BYTE CheckCC(const char *pszNumber)
{

```

```
#define HSE_VERSION    MAKELONG(HSE_
VERSION_MINOR, HSE_VERSION_MAJOR)
```

پس فایل Header فوق الذکر را در بالای فایل validate.cpp اضافه کنید. لطفاً توجه کنید! ما مقدار TRUE را بر می گردانیم تا به IIS اجازه استفاده از extension را بدهیم. قدم بعدی چیست؟ اکنون زمان آن رسیده است که روشی برای دریافت Query String از IIS بیابیم. اما چگونه؟ احتمالاً به خاطر می آورید که IIS از طریق اشاره گر به ECB می تواند با extension ما ارتباط برقرار کند؛ اشاره گری که به عنوان تنها پارامتر به تابع HttpExtensionProc پاس می شود:

```
DWORD WINAPI HttpExtensionProc(EXTEN
SION_CONTROL_BLOCK *pECB);
```

که در آن، pECB حاوی Data Member ای تحت عنوان lpszQueryString است که به Query String اشاره می کند. به بیان دیگر، اگر کاربر بخواهد از طریق این: URL: http://mydomain.com/validate.dll?12345، به extension ما دسترسی پیدا کند، pECB->lpszQueryString معادل ۱۲۳۴۵ خواهد بود، رشته ای که در دنباله علامت ؟ قرار گرفته است. مطلب دیگری که قبل از ادامه باید بر آن فائق بیابیم، آن است که بدانیم چگونه می توان از طریق pECB، رشته ای را به Client منعکس کرد (این رشته ممکن است HTML، Javascript و ... باشد). این مساله از طریق متد WriteClient موجود در بلوک ECB، امکان پذیر است که الگوی آن بدین شکل است:

```
BOOL (WINAPI *WriteClient)(HCONN
ConnID, LPVOID Buffer, LPDWORD
lpdwBytes, DWORD dwReserved);
```

که در آن، ConnID شناسه ارتباطی با Client ای است که پاسخ باید برایش ارسال شود، Buffer، به داده های ارسالی اشاره کرده، lpdwBytes طول Buffer مزبور بوده و dwReserved همانطور که از نامش بر می آید، کنار گذاشته می شود. با داشتن این اطلاعات اکنون می توانیم تابع HttpExtensionProc را پیاده سازی کنیم:

شروع پیاده سازی Extension

با در دست داشتن این اطلاعات، زمان آن رسیده تا extension خودمان را پیاده سازی کنیم. کامپایلر ++VC خود را اجرا کنید. از منوی فایل، گزینه New را انتخاب کنید. در حالی که Tab پروژه انتخاب شده است، بر روی گزینه Win32 Dynamic-Link Library کلیک سمت چپ Mouse را فشار دهید. در فضای "نام پروژه"، کلمه validate را نوشته و کلید ok را برای ادامه، فشار دهید. در پنجره Win32 Dynamic-Link Library، گزینه دوم یعنی A simple DLL project، را انتخاب کرده و کلید Finish را فشار دهید. اکنون، پروژه ای از نوع DLL داریم که آماده پیاده سازی است. از آنجایی که علاقه ای به پارامترهای ul_reason_for_call و hModule نداریم، آنها را نادیده می گیریم و به سادگی مقدار TRUE را باز می گردانیم. بنابراین تابع DLLMain ما به صورت زیر پیاده سازی می شود:

```
BOOL APIENTRY DllMain(HANDLE hMod-
ule,
    DWORD ul_reason_for_call,
    LPVOID lpReserved)
{
    return TRUE;
}
```

اکنون بیابید دنده را از DLLMain به اولین نقطه ورودی، GetExtensionVersion عوض کنیم. همانطور که به یاد می آورید، ما این تابع را به منظور فراهم کردن دو مطلب (که قبلاً توضیح داده شد) پیاده سازی می کنیم. پس بیابید تا آنرا به پروژه بیفزاییم:

```
BOOL WINAPI GetExtensionVersion(HSE_
VERSION_INFO *pVer)
{
    pVer->dwExtensionVersion = HSE_
VERSION;
    strncpy(pVer->lpszExtensionDesc,
        "Validate ISAPI Extension",
        HSE_MAX_EXT_DLL_NAME_LEN);
    return TRUE;
}
```

که در آن، HSE_VERSION، در فایل هدر httpext.h به صورت زیر تعریف شده است:

```

DWORD WINAPI HttpExtensionProc(EXTENSION_CONTROL_BLOCK *pECB)
{
    StartContext(pECB);

    BYTE byRet = CheckCC(pECB->lpszQueryString);
    if(!byRet)
    {
        //this is a valid master card, echo a suitable string to the client
        WriteContext(pECB,
            "<p><b><font face='Verdana'
            color='#008000'>Congratulations!</font></b></p>");
        WriteContext(pECB,
            "</b></p><p><font size='2' face='Verdana'>");
        WriteContext(pECB,
            "%s is a valid master card #</font></p>\r\n",
            pECB->lpszQueryString);
    }
    else
    {
        //this is an invalid master card, echo a proper string to the client!
        WriteContext(pECB,
            "<p><b><font face='Verdana'
            color='#800000'>Sorry!</font></b></p>");
        WriteContext(pECB,
            "<p><font size='2' face='Verdana'>What you have entered is an ");
        WriteContext(pECB,
            "invalid master card#</font></p>\r\n");
    }

    EndContext(pECB);

    return HSE_STATUS_SUCCESS;
}

```

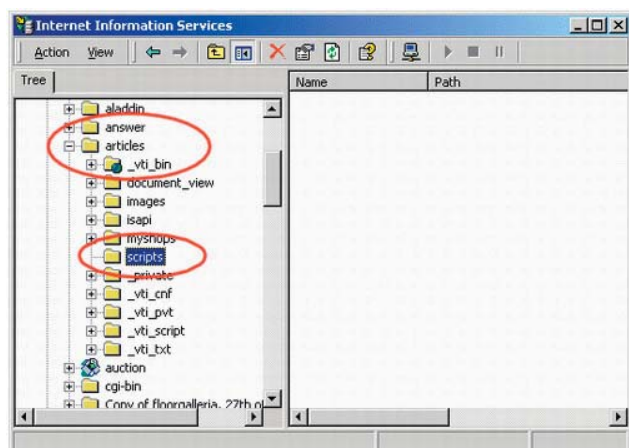
و توابع، StartContext، EndContext و WriteContext چیستند؟ تابع WriteContext، تابعی است که عمل ارسال یک رشته به Client را ساده می کند و به صورت زیر پیاده سازی شده است:

```

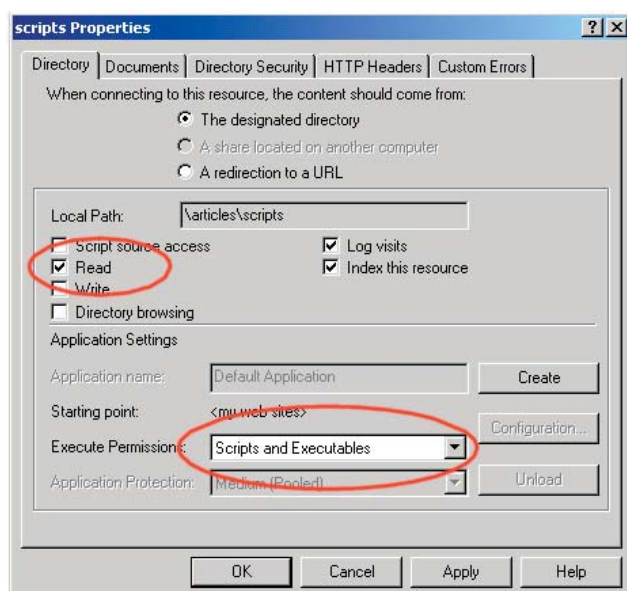
void WriteContext(EXTENSION_CONTROL_BLOCK *pECB, char *pszFormat, ...)
{
    char szBuffer[1024];
    va_list arg_ptr;
    va_start(arg_ptr, pszFormat);
    vsprintf(szBuffer, pszFormat, arg_ptr);
    va_end(arg_ptr);

    DWORD dwSize = strlen(szBuffer);
    pECB->WriteClient(pECB->ConnID, szBuffer, &dwSize, 0);
}

```

بر روی شاخه scripts، کلید سمت راست Mouse را فشار داده، گزینه Properties را انتخاب کنید.



تغییرات لازم را در پنجره اعمال کنید تا شبیه شکل فوق به نظر برسد. سپس، کلید Apply را به منظور اعمال تغییرات، فشار دهید. اکنون فایل validate.dll را، از شاخه release به درون شاخه scripts ای که هم اکنون پیکربندی کردید، کپی کنید. مرورگر مورد علاقه تان را باز کنید و سعی کنید تا در حالی که عددی در قالب Query String به DLL پاس می کنید، به آن دسترسی پیدا کنید:

<http://myth/articles/scripts/validate.dll?1234567890125436>

لطفاً دقت کنید که دامنه URL فوق را بمنظور همسو سازی با شرایط اجرایی خودتان، تغییر دهید. بدین ترتیب، صفحه نام آشنای نفرین شده زیر را خواهید دید:

بدین ترتیب دیگر نیازی به ارسال اطلاعات تکراری و بیپهوده همانند طول رشته، هنگام ارسال یک رشته به Client نخواهد بود. علاوه بر این، می توانیم از این تابع همانند متود CString::Format() موجود در MFC، استفاده کنیم:

```
WriteContext(pECB, "5 + 6 = %d", 5 + 6);
```

که باعث ارسال $5 + 6 = 11$ به Client می شود. از طرف دیگر، StartContext، به منظور ارسال اطلاعات Heading یا هرگونه اطلاعات آغاز شونده در کد HTML، پیاده سازی شده است. همتای این تابع، EndContext، پاورقی HTML را به Client ارسال می کند:

```
void StartContext(EXTENSION_CONTROL_BLOCK *pECB)
{
    WriteContext(pECB, "<html>\r\n<body>\r\n");
}

void EndContext(EXTENSION_CONTROL_BLOCK *pECB)
{
    WriteContext(pECB, "</body>\r\n</html>");
}
```

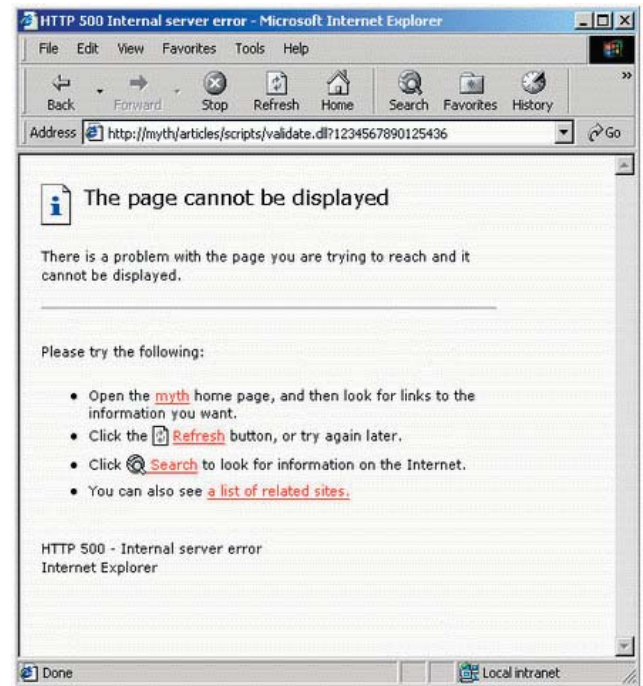
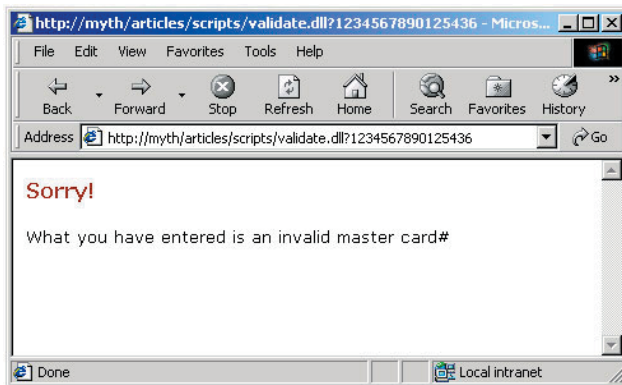
حال، پیکر بندی پروژه فعال را به Release تغییر داده و برنامه را کامپایل کنید.

اکنون زمان آن رسیده تا ببینیم چگونه extension خود را در IIS نصب کرده و از آن استفاده کنیم.

با این فرض که شما در حال اجرای Windows 2000 هستید، گزینه Internet Services Manager را از منوی Administrative Tools در منوی Programs اجرا کنید، تا IIS snap-in اجرا شود. از بخش سمت چپ، مسیرهای دلخواه خود را که می خواهید extension تازه پیاده سازی شده را در آن اجرا کنید، انتخاب نمایید.

معمولاً این مساله در شاخه scripts دامنه مورد نظر، رخ می دهد. بدین منظور، من شاخه ای تحت عنوان scripts در زیر شاخه articles ایجاد کرده ام:

سپس از منوی پروژه، گزینه add to project و سپس گزینه files را انتخاب کنید. فایل جدید ایجاد شده def را به پروژه بیفزایید. مجدداً، DLL را کامپایل کنید. سپس فایل را از شاخه release به شاخه scripts کپی کرده و صفحه مزبور را مجدداً مرور کنید. این بار، صفحه زیر را دریافت خواهید کرد:



اکنون، همین صفحه را با شماره کارت اعتباری معتبری به روز کنید و صفحه تیریکی را خواهید دید. سپس، فایل scripts.dll موجود در شاخه scripts را حذف کنید. چه اتفاقی افتاد؟ درست است. خطایی مبنی بر در حال استفاده بودن DLL خواهید گرفت و نمی توانید آنرا حذف کنید. پس چگونه فایل را به روز کنیم، اگر آنرا تغییر دادیم؟ آیا باید کامپیوتر را Restart کنیم، Client را ببندیم، یا ... خیر. هیچ یک!

تمام کاری که باید انجام دهید، متوقف کردن IIS است. این مساله از طریق IIS snap-in میسر است. آن را اجرا کرده و کلید سمت راست Mouse را روی نام سرور فشار دهید.

از منوی باز شده، restart IIS را انتخاب کنید. در پنجره، "می خواهید IIS چه کار کند؟"، گزینه stop Internet services را انتخاب کرده، کلید ok را فشار دهید. چند ثانیه مکث کنید و انجام شده است! اکنون می توانید extension خود را تغییر داده، یا آنرا حذف کنید. جالب بود، اینطور نیست؟

سخن نهایی

این تمام کاری است که ISAPI Extension ها انجام می دهند. آن ها کارایی IIS را توسعه می دهند. در این مقاله، من عبارت extension را به طور مداوم تکرار کردم.

اکنون زمان آن است که بگویم برنامه های ISAPI به دو دسته تقسیم می شوند: ISAPI Extensions و ISAPI Filters. فیلترهای ISAPI، بر خلاف Extension ها، به ازای هر درخواست از وب سرور فراخوانی می شوند. به عبارت دیگر، آن ها سرعت وب

وای! اشتباه چه بوده که این صفحه نمایان شده است؟ اگر اندکی تفکر کنید، احتمالاً به خاطر خواهید آورد که باید سه تابع مورد استفاده در برنامه را در معرض دید IIS قرار می دادیم (در مورد مثال ما، دو تابع)، GetExtensionVersion و HttpExtensionProc.

پس اجازه دهید برای رفع این مساله به پروژه بازگردیم. هنگامی که در محیط VC++ هستید، فایل جدیدی تحت عنوان validate.def ایجاد کنید و این فایل را بدین صورت پیاده سازی کنید:

```
; Validate.def : Declares the module
parameters for the DLL.
```

```
LIBRARY "Validate"
DESCRIPTION 'Validate ISAPI Extension'
```

```
EXPORTS
```

```
; Explicit exports can go here
```

```
HttpExtensionProc @1
GetExtensionVersion @2
```


سرور را به طرز چشمگیری کاهش می دهند، زیرا مدام در حال فراخوانی هستند. اما آن ها هنگام ایجاد سرویس های Logging یا اموری از این دست، مفیدند. از آنجایی که توضیح ISAPI Filter ها مقاله دیگری را می طلبد، اجازه دهید تا این مطلب را به شما واگذار کنم تا از چند و چون کارکرد آن ها مطلع شوید.

در هر حال، این ساده ترین ISAPI Extension ای بود که امروز پیاده سازی کردیم تا به شما نشان دهیم که یک extension چیست و چگونه کار می کند. این extension، یک DLL ساده تک Thread ای بود، که ساده ترین گونه DLL ها برای پیاده سازی است. در صورتی که برنامه های واقعی و درجه اول به این سادگی ها قابل پیاده سازی نیستند، زیرا باید با مسائلی از قبیل: multi-threaded، connection pooling و دیگر عناوین پیشرفته مواجه شوند.

این مساله به عهده شما است که بیاموزید چگونه می توانید آن ها را به خوبی بنوازید و این مقاله فقط نقطه شروع است. همین بود، دوستان. بدرود.


منابع:

- Professional Visual C++ ISAPI Programming, by Michael Tracy
- Microsoft Developer Network, MSDN



مجله الکترونیکی برنامه نویسی

متفاوت ببینید، متفاوت تبلیغ کنید....



مجله برنامه نویسی

اولین و تخصصی ترین مجله الکترونیکی در زمینه علوم نرم افزار

تلفن سازمان آکمی ها:

۰۲۱ ۷۷ ۸۴ ۷۳ ۶۴

۰۲۱ ۷۷ ۲۶ ۹۴ ۷۳

جهت کسب اطلاعات بیشتر به
آدرس اینترنتی زیر مراجعه فرمایید:

<http://adv.barnamenevis.org>

قابلیت‌های جدید ASP.NET 4.0

behrouz.rad@gmail.com

مقدمه: در این ستون سعی می‌کنم در هر شماره باختصار چند مورد از قابلیت‌های جدید ASP.NET 4.0 را معرفی کنم. امید است مورد توجه علاقه‌مندان قرار گیرد.

طریق الگوریتم GZip صورت می‌گیرد. بدین‌منظور، خاصیت `compressionEnabled` تگ `sessionState` را در فایل `Web.Config` به `true` تنظیم کنید. از آن‌جا که فشرده‌سازی داده‌ها و بالطبع، خارج ساختن آن‌ها از حالت فشرده، سربار به برنامه تحمیل می‌کند، فشرده‌سازی را فقط برای داده‌های با حجم بالا فعال کنید.

متدهای MetaDescription و MetaKeywords

ASP.NET 4.0 در بحث SEO پیشرفت‌های خوبی داشته‌است. دیگر نیاز نیست تا با روش‌های نامتعارف، تگ‌های `meta` ی `description` و `keywords` را به صفحه اضافه کنید. متدهای `MetaDescription` و `MetaKeywords` کلاس `Page`، تگ‌های `meta` ی `description` و `keywords` را به صفحه اضافه می‌کنند.

- مستندات متد `MetaDescription` در MSDN

[http://msdn.microsoft.com/en-us/library/system.web.ui.page.metadescription\(vs.100\).aspx](http://msdn.microsoft.com/en-us/library/system.web.ui.page.metadescription(vs.100).aspx)

- مستندات متد `MetaKeywords` در MSDN

[http://brad.barnamenevis.org/msdn.microsoft.com/en-us/library/system.web.ui.page.metadescription\(vs.100\).aspx](http://brad.barnamenevis.org/msdn.microsoft.com/en-us/library/system.web.ui.page.metadescription(vs.100).aspx)

متد `RedirectPermanent`

متد `Redirect` کلاس `Response` مشکل بزرگی دارد و آن هم این‌که که کد وضعیت ۳۰۲ را برمی‌گرداند. کد وضعیت ۳۰۲ نمایانگر انتقال "موقت" صفحه‌ی وب هست که هم در کاهش SEO و هم در افزایش `Round-trip` به سرور مؤثر هست. افرادی که خود اقدام به ساختن مازول‌های `URL Rewriting` می‌کردند، با دو مشکل فوق مواجه بودند. در ASP.NET 4.0، متد `RedirectPermanent` به کلاس `Response` اضافه شده‌است که کد ۳۰۱ را که بیانگر انتقال "دائمی" منبع مورد نظر هست برمی‌گرداند. در این حالت، `URL` دائمی در هدر `Location` قرار می‌گیرد و موتور جستجو به راحتی صفحه‌ی جدید را ایندکس می‌کند. همچنین از ایجاد `Round-trip` اضافه به سرور نیز جلوگیری خواهد شد.

فشرده‌سازی Session

`Session` ها به طور پیش‌فرض در حافظه‌ی سرور نگهداشته می‌شوند. `Session` ها را می‌توان در جای دیگر مثلاً در بانک‌ی `SQL Server` نیز ذخیره کرد. اگر حجم داده‌های ذخیره شده در `Session` زیاد باشد، باعث افزایش حجم پایگاه داده خواهد شد. ASP.NET 4.0 قابلیت جدیدی را معرفی کرده که از طریق آن می‌توان `Session` ها را فشرده کرد. این فشرده‌سازی از

تعریف ویژگی‌ها (Properties) در جاوا اسکریپت

salar@softprojects.org

مقدمه: یکی از امکانات نسخه‌های جدید JavaScript پشتیبانی از ویژگی‌ها (Property) هست. اگر با سایر زبان‌های برنامه‌نویسی آشنایی داشته باشید، مطمئناً می‌دانید که ویژگی‌ها یکی از بخش‌های جدایی ناپذیر زبان‌های برنامه‌نویسی OOP هستند. ویژگی‌ها این امکان را فراهم می‌کنند که بر روی مقادیر متغیرهای خود کنترل داشته باشید و حق دسترسی برای آن‌ها جهت ویرایش تعیین کنید. همچنین می‌توان داده‌های وارده را فیلتر کرده تا فقط محدوده و داده‌های مورد نظر اعمال شوند. با این مقدمه می‌خواهم چند روش تعریف ویژگی‌ها در جاوا اسکریپت مطرح کنم.

روش اول:

جاوا اسکریپت زبان بسیار ساده‌ای است. آبجکت‌ها (یا اشیا) در جاوا اسکریپت چیزی بیش از مجموعه‌ای از کلیدها و مقادیر نیستند. به این معنی که برای دسترسی به عضوی از یک شیء می‌توانید همانند یک آرایه با آن رفتار کنید. برای مثال برای افزودن یک عضو به یک شیء به مثال زیر توجه کنید:

```
var thing = new Object();
thing[ "prop1" ] = "Hello";
```

در این مثال ما عضو جدیدی به نام prop1 را به thing اضافه کردیم. مقدار prop1 برابر Hello خواهد بود. روش دیگری هم برای افزودن عضو به شیء مورد وجود دارد. مثال:

```
var thing = new Object();
thing.prop1 = "Hello";
```

روش دیگر برای معرفی اشیا استفاده از لیترات‌های آبجکت هست که نتیجه‌ای دقیقاً مانند دو مثال قبلی دارد:

```
var thing = {prop1 : "Hello"};
```

استفاده از مقدار prop1 بسیار ساده است می‌توان همانند یک آرایه یا یک ویژگی از آن استفاده کرد:

```
var thing = new Object();
thing.prop1 = "Hello";
alert( thing["prop1"] );
thing.prop1 = "Hello Again!";
alert( thing.prop1 );
```

روش دوم:

روشی که در بالا توضیح دادم خیلی ساده و کاربردی است ولی تنها محدودیتی که دارد این است که امکان کنترل بر داده‌هایی که قبول می‌کند و یا مقداری که باید برگرداند وجود ندارد. برای رفع این مشکل شرکت Mozilla راه حلی را ارائه کرده است که به مرور سایر مرورگرها نیز از آن پشتیبانی می‌کنند. این روش چیزی جز کلمات کلیدی get و set نیست.

```
var thing = {
  _price: 0,
  get price() { return this._price;
  },
  set price(value)
  {
    if (value < 0)
      throw "price must be greater
      than zero";
    this._price = value;
  }
};
```

در این مثال شیء با نام thing تعریف شده است. این شیء دارای یک متغیر با نام _price و ویژگی price هست که با استفاده از یک متد get و یک متد set تعریف شده است. به طور ساده‌تر زمانی که می‌خواهید مقداری را از ویژگی price بخوانید تابع معرفی شده در مقابل get فراخوانی خواهد شد و همچنین، هنگام مقدار دهی به price تابع معرفی شده در مقابل set فراخوانی خواهد شد. همانطور که می‌بینید ویژگی price در هنگام عمل set مقدار ورودی را بررسی می‌کند و اگر کوچک‌تر از صفر باشد خطایی را تولید خواهد کرد. نمونه استفاده از کد بالا:

```
// مقدار دهی
thing.price = 5;
// نمایش مقدار
alert( thing.price );
// مقدار نامعتبر و تولید خطا
thing.price = -10;
```

روش چهارم:

در این روشی که می‌خواه معرفی کنیم، واقعاً ویژگی‌ها را پیاده سازی نمی‌کند ولی مانند یک ویژگی عمل می‌کند و می‌تونه در کار شما مفید واقع بشه. در این روش در حقیقت ما دو تابع جداگانه برای خواندن و نوشتن در متغیر تعریف می‌کنیم و کارهای مورد نظر را در داخل آن‌ها انجام می‌دهیم.

```
function thing()
{
    var _price = 0;
    this.getPrice = function() {
        return _price;
    };
    this.setPrice = function(val) {
        if (val < 0)
            throw "price must
            be greater than 0";
        _price = val;
    };
}
```

در این مثال شیء **thing** به عنوان یک تابع تعریف شده است که می‌توان از روی متغیر و اشیا دیگری هم ساخت. در داخل این تابع دو تابع **getPrice** برای خواندن و **setPrice** برای نوشتن در متغیر **price** تعریف شده است.

```
// ایجاد نمونه جدید از تابع تعریف شده
var thing = new thing();
// دسترسی متغیر داخلی
alert(product._price);
// مقدار دهی با استفاده از تابع
product.setPrice(2);
// دسترسی به مقدار با استفاده از تابع مربوطه
alert( product.getPrice() );
// مقدار با یک مقدار نامعتبر و دریافت خطا
product.setPrice( -4 );
```

با این اوصاف مشاهده می‌کنید که **javascript** روز به روز قوی‌تر می‌شود، مخصوصاً که **Web 2.0** بسیار به **javascript** وابسته هستند.

```
// مقدار دهی
thing.price = 5;
// نمایش مقدار
alert( thing.price );
// مقدار نامعتبر و تولید خطا
thing.price = -10;
```

متأسفانه از این روش مرورگرهای کمی پشتیبانی می‌کنند و در حال حاضر مرورگرهای **Firefox** و **Safari 3** و **Opera 9.5** از آن پشتیبانی می‌کنند.

روش سوم:

این روش را مرورگر **Firefox** برای استفاده در خودش طراحی کرده و در سایر مرورگرها کار نخواهد کرد. روش بسیار جالبی است و ساده تر از روش قبلی می‌باشد.

```
var thing =
{
    _price: 0
};
thing.__defineGetter__("price", function() { return this._price; });
thing.__defineSetter__("price", function(value)
{
    if (value < 0)
        throw "price must be
        greater than 0";
    this._price = value;
});
```

در این مثال ابتدا شیء **thing** تعریف می‌شود. سپس با استفاده از تابع **defineGetter** متد **get** با برای ویژگی **price** تعریف می‌شود. این تابع دو ورودی دارد که اولی نام ویژگی مورد نظر است و دومی رفرنسی به تابع مورد نظر برای **get**. البته در این مثال تابع در همان جا تعریف شده است.

تابع **__defineSetter** همانند **__defineGetter** عمل می‌کند و فقط برای اضافه کردن تعریف **set** به ویژگی مورد نظر مورد استفاده است. در این مثال تابع مربوط به **set** در همان خط تعریف شده و مقدار ورودی را برای مقادیر کمتر از صفر کنترل می‌کند.

همانند مثال قبلی این مثال نیز کار خواهد کرد:

آموزش زبان برنامه نویسی F# بخش دوم

mehdi.asgari@yahoo.com



مقدمه: در قسمت قبل نکات ابتدایی این زبان را یاد گرفتیم. در این قسمت به ادامه معرفی ویژگی‌های این زبان می‌پردازیم. (همانند شماره قبل برای تمام مثال‌ها معادل سی شارپ را نیز خواهیم آورد)

توسط یک extension method این قابلیت را به کلاس Func اضافه کرد که در اینجا به آن نمی‌پردازم؛ برای توضیحات بیشتر برای سی شارپ لینک زیر را ببینید: <http://blogs.msdn.com/wesdyer/archive/2007/01/29/currying-and-partial-function-application.aspx> فراخوانی زیر:

```
let result = mul7 8
```

با فراخوانی زیر برابر است:

```
let result = multiply 7 8
```

(انگار به جای mul7 عبارت "multiply 7" را جایگزین کرده ایم. تابع mul7 تابع multiply را به صورت ناقص و با یک پارامتر فراخوانی کرده است، پس نیاز به یک پارامتر دیگر هم دارد تا فراخوانی multiply کامل شود)

: Higher Order Function

به تابعی که ورودی و/یا خروجی آن تابع باشد، یک تابع HOF گفته می‌شود. در مثال قبل تابع mul7 یک HOF است زیرا یک تابع برمی‌گرداند. همچنین توابع map و fold و reduce (که در این مقاله با آن‌ها آشنا خواهیم شد) که یکی از ورودی‌هایشان تابع است، HOF هستند.

لیست در F#:

شاید تعجب کنید که چرا با لیست شروع کردم. چرا مثلاً با آرایه شروع نکردم؟ لیست از اساسی‌ترین ساختار داده‌های مورد استفاده در زبان‌های تابعی است که کاربردی بیشتر از آرایه دارد. لیست‌ها

قبل از پرداختن به ویژگی‌های خود زبان F#، چند واژه را که در دنیای برنامه‌نویسی تابعی زیاد به گوش‌تان خواهد خورد توضیح می‌دهم:

: Partial Application

به فراخوانی ناقص یک تابع گفته می‌شود. یعنی مثلاً تابعی را که سه پارامتر دارد، با یک یا دو پارامتر فراخوانی کنیم. این کار برای کسی که با زبان‌های تابعی برنامه‌نویسی نکرده باشید غیرعادی به نظر می‌رسد. مثال:

تابع زیر ۲ عدد را در هم ضرب کرده و نتیجه را بر می‌گرداند:

```
let multiply x y = x * y // F#

Func<int, int, int> multiply = (x, y) => x * y; // C#
```

اگر شماره قبل این مقاله را خوانده باشید می‌توانید حدس بزنید که امضای این تابع به شکل

```
int -> int -> int
```

است (یعنی دو int گرفته و یک int بر می‌گرداند).

حالا اگر تابع را به این شکل فراخوانی کنیم:

```
let mul7 = multiply 7
```

یک تابع جدید با نام mul7 و با امضای (int -> int) داریم؛ یعنی مقدار خروجی این تابع خود تابعی است که یک int گرفته و یک int بر می‌گرداند. (C# چنین چیزی به صورت آماده ندارد و باید

دستیابی به اعضای آرایه و لیست (چاپ عنصر ششم):

```
open System
Console.WriteLine myArray1[5]
```

معادل در سی شارپ:

```
using System;
int[] myArray1 = { 1, 2, 3, 4, 5, 6,
7, 8, 9, 10 };
Console.WriteLine(myArray1[5]);
```

همانطور که می بینید تنها تفاوت در نقطه ای است که قبل از کروش قرار دارد.

Tuple: تاپل یا چندتایی نوعی ساختمان داده است که شامل چند (معمولاً بیش از یک) عضو است که اعضا لزوماً از یک نوع نیستند. **Tuple** ها **immutable** (تغییر ناپذیر) هستند. اعضای یک **tuple** توسط کاما از یکدیگر جدا می شوند و معمولاً داخل یک پرانتز قرار می گیرند (پرانتز الزامی نیست ولی بهتر است باشد تا مجموعه بودن اعضا را نشان دهد). **C#** چنین مفهومی ندارد؛ البته در **NET 4.0** که الان در نسخه بتاست، کلاس **Tuple** جزئی از **framework** خواهد بود. تقریباً تمام زبان های تابعی و زبان های داینامیک مفهوم **tuple** را پشتیبانی می کنند، ولی در زبان های متداول تر (مثل **C#**) این ساختار جزئی از زبان نیست و توسط کلاس هایی ارائه می شود.

مثال:

```
let person = (12, "Mehdi", 15.67)
let four_numbers = 2,3,5,7
```

خب ببینیم نوع این مقادیری که تعریف کردیم چیست. نوع **person** **int * string * float** (توجه کنید که از * استفاده شده؛ با -> که در توابع استفاده می شود قاطی نکنید). نوع **four_persons** **int * int * int * int** فرض کنید بخواهیم نام یک شخص را از **person** استخراج کنیم. ("Mehdi" در مثال بالا)

```
let _, name, _ = person
```

عملی که انجام دادیم نوعی **pattern matching** ساده است (با **pattern matching** که از اجزای زبان های تابعی و یکی از نقاط برتری آن ها بر دیگر زبان هاست بعداً مفصل آشنا خواهیم شد). در

در **F#** غیر قابل تغییر هستند (**immutable**) مثل رشته؛ یعنی در صورت حذف یک عنصر از لیست، یا افزودن دو لیست به هم، لیست جدیدی حاصل خواهد شد و لیست اصلی دست نخورده باقی می ماند. تعریف لیست در **F#**:

```
let myList1 = [1..10]
let myList2 = [1; 2; 7; -98; 0; 9]
```

خط اول اعداد ۱ تا ۱۰ است. مشابه در سی شارپ:

```
var myList1 = new List<int>() {
1,2,3,4,5,6,7,8,9,10};
var myList2 = new List<int>() {
1,2,7,-98,0,9};
```

همانطور که می بینید لیست مانند اعداد و رشته ها جزو ساختارهای اصلی زبان **F#** است. (در **F#** برای جدا کردن اعضای لیست یا آرایه از '؛' استفاده می شود نه از کاما. کاما در **tuple** ها استفاده می شود که در ادامه خواهیم دید). برای چسباندن دو لیست به یکدیگر و ایجاد لیستی جدید شامل دو لیست اول از عملگر **@** استفاده می شود:

```
let myArray1 = [ 1..10 ]
let myArray2 = [ 20..30 ]
let myArray3 = myArray1 @ myArray2
```

لیست سوم شامل اعداد ۱ تا ۱۰ و ۲۰ تا ۳۰ است.

آرایه

آرایه نیز همانند لیست تعریف می شود، تنها تفاوت در دو خط عمودی است که در ابتدا و انتهای مجموعه اعضا قرار می گیرد. مثال:

```
let myArray1 = [| 1..10 |]
let myArray2 = [| "Hello"; "World";
"Of"; "F#" |]
```

آرایه اول مانند لیست اول مثال قبل است و آرایه دوم شامل چهار رشته است. مشابه در سی شارپ:

```
int[] myArray1 = { 1, 2, 3, 4, 5, 6,
7, 8, 9, 10 };
string[] myArray2 = { "Hello",
"World", "Of", "F#" };
```



```
let s = "123"
let (result,value) = System.Int32.
TryParse(s)
match result with
| true -> printf "%d\n" value
| false -> printf "invalid string
format\n"
```

دیکشنری:

دیکشنری یا map یا key-value pair نوعی ساختمان داده است که اعضای آن دوتایی هستند: یک کلید و یک مقدار؛ که کلیدها نگاشت می‌شوند به مجموعه مقادیر. در سی‌شارپ از HashTable و Dictionary برای این کار استفاده می‌شود. F# دو نوع دیکشنری دارد: Map و dict

```
let myDictionary = dict [
(1,"Mehdi"); (2,"Ali"); (7,"Saeed")
]
```

نوع مقدار فوق Dictionary<int,string> است (کلاس System.Collections.Generic.Dictionary)

```
let myMap = Map [ ("A","Mehdi");
("B","Ali"); ("C","Saeed") ]
```

Map از کلاس‌های خود F# است. نوع مقدار بالا Map<string,string> است. برای دسترسی به مقادیر دیکشنری نیز همانند لیست و آرایه از شکل زیر استفاده می‌شود:

```
let A = myMap.[ "A" ]
```

ادامه بحث توابع:

• در F# برای برگرداندن مقدار در یک تابع، کلمه return وجود ندارد، بلکه نوع آخرین خط، مشخص کننده مقدار برگشتی تابع است. مثال:

مقدار برگشتی تابع زیر int است: (امضا: int -> int)

```
let myFunc arg1 =
arg1 + 10
```

مثال بالا می‌گوییم که مقدار person شامل ۳ عضو است، منتها ما با عضو اول و سوم کار نداریم (با استفاده از underscore مشخص می‌کنیم که این مقادیر برایمان اهمیتی ندارد) اما مقدار دوم را به نام name منتسب می‌کنیم. پس name حاوی رشته "Mehdi" خواهد بود.

متدهایی در دات نت وجود دارند که دو مقدار برمی‌گردانند: یکی از طریق مقدار برگشتی تابع و دیگری از طریق یک پارامتر out (مثل تابع TryParse کلاس Int32). در صورت true بودن مقدار برگشتی متد، آرگومان مذکور شامل مقدار مورد نظر خواهد بود، وگرنه متد موفقیت آمیز به پایان نرسیده (این روش در برخی از مواقع بهتر از ایجاد و پرتاب یک exception است) مثال از سی‌شارپ:

```
string s = "123";
int value ;
if (Int32.TryParse(s, out value))
{
    Console.WriteLine("value is {0}",
value);
}
else
{
    Console.WriteLine("invalid string
format");
}
```

در F# چنین متدهایی یک tuple برمی‌گردانند که دو عضو دارد: عضو اول از نوع bool که نشان دهنده موفقیت آمیز بودن نتیجه و عضو دوم نیز مقدار مورد نظر است. مثال بالا در F#:

```
let s = "123"
let (result,value) = System.Int32.
TryParse(s)
if result then
    printf "%d\n" value
else
    printf "invalid string format\n"
```

همین مثال توسط pattern matching (توضیح pattern matching) بهمانند برای شماره بعد. فعلاً فقط همین قدر بگوییم که مفهوم مشابه در زبان‌های دیگر عبارت switch-case است منتها با قدرت و انعطاف‌پذیری بسیار کمتر از pattern matching:

• توابع لامبدا (closure): توابعی که درون توابع دیگر تعریف می شوند. (این کار از طریق anonymous methods در C# 2.0 و توابع لامبدا در C# 3.0 قابل انجام است) مثال:

```
let outer_func m =
    let pi = 3.14
    let inner_func n =
        // do some stuff here
        n * n * pi
    inner_func m
```

درون تابع outer_func یک تابع با نام inner_func ایجاد کرده ایم. مشابه در C#:

```
static float outer_func(float m)
{
    float pi = 3.14F;
    Func<float, float> inner_func =
    (n) =>
    {
        // do stuff here
        return n * n * pi;
    };
    return inner_func(m);
}
```

مثال دیگر: توابع یک خطی که معمولاً در آرگومان توابع استفاده می شوند:

```
let squares = List.iter( fun x ->
    printf "%d " x) [1..10]
```

در این مثال با استفاده از متد iter ماژول List (از ماژول های استاندارد F# که بعداً بررسی خواهیم کرد) بر روی تک تک اعضای یک لیست عملی را انجام می دهیم (در اینجا اعداد ۱ تا ۱۰ را که اعضای لیست مورد نظر هستند در کنسول چاپ می کنیم). در واقع تابع iter دو آرگومان دریافت می کند: آرگومان اول تابعی است که یک عنصر را گرفته و کاری انجام می دهد (ولی چیزی بر نمی گرداند) و آرگومان دوم لیستی از عناصر است. امضای تابع iter به این شکل است:

“T list -> unit” -> “T -> unit”

آرگومان هایی که نوعشان همراه با single quote (‘) همراه است، آرگومان های جنریک هستند (یعنی می توانند از هر نوعی باشند)،

مقدار برگشتی تابع زیر unit است (مشابه void در سی شارپ). (امضا: int -> unit)

```
let myFunc arg1 =
    printf "%d" arg1
```

توجه داشته باشید که تابع printf چیزی بر نمی گرداند (فقط کاری انجام می دهد) مقدار برگشتی تابع زیر float است:

```
let myFunc x y z =
    let w = x * y
    let z = sqrt(z)
    let result = (w + z) / 2.0
    result
```

• **توابع بازگشتی:** در F# بر خلاف اغلب زبان های متداول (مثل سی شارپ) تعریف یک تابع بازگشتی با یک تابع معمولی فرق می کند. توابع بازگشتی با rec (مخفف recursive) مشخص می شوند. به عنوان مثال کد زیر (تابع فاکتوریل) اشتباه است:

```
let factorial n =
    if n = 1 then 1 else n *
    factorial(n - 1)
```

کد زیر اصلاح شده کد فوق است:

```
let rec factorial n =
    if n = 1 then 1 else n *
    factorial(n - 1)
```

در زبان های تابعی معمولاً به جای if-then-else از pattern matching و به جای for و while از توابع بازگشتی استفاده می شود. سوالی که ممکن است مطرح شود این است که آیا استفاده از توابع بازگشتی به جای حلقه، باعث کاهش سرعت و بهینه نبودن کد F# نمی شود؟

درست است که در اکثر زبان ها برای فراخوانی توابع از پشت به پشت استفاده می شود، ولی در زبان های تابعی از بهینه سازی ای موسوم به tail call optimization استفاده می شود تا از سربار ایجاد پشت به جلوگیری شود (در درس های آتی مفصل تر به این موضوع خواهیم پرداخت)

نتیجه عدد ۵۵ است (جمع اعداد ۱ تا ۱۰)

تابع `filter`: این تابع امضایی به شکل زیر دارد:

`(T -> bool) -> 'T list -> 'T list`

این تابع دو آرگومان دارد: یک تابع و یک لیست. تابع آرگومان، یک عدد گرفته و `true` یا `false` بر می گرداند. در نهایت لیستی برگردانده می شود که شامل اعضای است که اعمال تابع آرگومان بر روی آن ها نتیجه `true` بر می گرداند. مثال: استخراج اعداد فرد از لیست اعداد

```
let odds = List.filter (fun x -> x % 2 <> 0) [1..10]
```

نتیجه: لیستی با اعضای [۱؛۳؛۵؛۷؛۹]

• عملگر `>`

از این عملگر برای تعویض جای تابع و آرگومان ها استفاده می شود. فرض کنید می خواهیم مجموع مربع اعداد زوج بین ۱ تا ۵۰ را چاپ کنیم؛ روش اول که به ذهنمان می رسد احتمالاً بدین شکل است:

```
let numbers = [1..50]
let mutable sum = 0
for i in 1..49 do
    let n = numbers.[i]
    if n % 2 = 0 then sum <- sum + (n * n)
printf "%d" sum
```

روش تابعی تر استفاده از توابع مخصوص زبان های تابعی است که در قسمت قبل با آن ها آشنا شدیم:

```
printf "%d" (List.reduce (fun x y -> x + y) (List.map (fun x -> x * x) (List.filter (fun x -> x % 2 = 0) [1..50])))
```

همانطور که می بینید اصلاً کد خوانایی نیست و خود من که کد را نوشته ام باید چند دقیقه به آن نگاه کنم تا متوجه بشوم این کد چکار می کند.

حالا همین کد را با استفاده از `>` بازنویسی می کنیم:

پس تابع `iter` بر روی هر نوع لیستی عمل می کند. مشابه در `#C`:

```
var numbers = new
List<int>(Enumerable.Range(1, 10));
numbers.ForEach(x => Console.
WriteLine(x));
```

• `reduce`، `map` و `filter`: این ۳ تابع از توابع خیلی مهم در تمام زبان های تابعی هستند که با اهمیتشان به مرور و در خلال مثال ها آشنا خواهیم شد. این توابع برای اکثر ساختمان داده ها موجودند ولی ما در اینجا فقط با نسخه `List` آن ها سر و کار داریم (مهم درک مفهوم این توابع است). تابع `map` امضایی به این شکل دارد:

`(T -> 'U) -> 'T list -> 'U list`

این تابع یک تابع را می گیرد و آن را بر روی اعضای یک لیست اعمال کرده و یک لیست جدید بر می گرداند (لیست اصلی دست نخورده باقی می ماند). مثال: سه برابر کردن تمام اعضای یک لیست:

```
let numbers = List.map (fun x -> x * 3) [1..10]
```

نتیجه یک لیست است با اعداد ۳ تا ۳۰ (۳، ۶، ۹، ...، ۳۰) مثال دیگر: تبدیل تمام اعضای یک `int list` به رشته:

```
let strings = List.map (fun x -> x.ToString()) [1..10]
```

تابع `reduce` عملی را بر روی تمام اعضای یک لیست انجام داده و در نهایت یک مقدار نهایی بر می گرداند. امضای تابع:

`(T -> 'T -> 'T) -> 'T list -> 'T`

مثال: جمع تمام اعضای یک لیست با هم: در واقع عمل جمع دو به دو روی اعضا انجام شده و در نهایت یک عدد برگردانده می شود.

```
let sum = List.reduce (+) [1..10]
```

الگوریتم ساده فیبوناچی :

```
let rec fibo n =
  if n = 2 or n = 1 then 1 else
  fibo (n - 1) + fibo (n - 2)
```

محاسبه عدد ۴۴ام با این روش روی کامپیوتر من (پردازنده دو هسته‌ای ۱.۸ GHZ) دوازده ثانیه طول کشید. (اگر حوصله دارید اعداد بالاتر را امتحان کنید) علت کند بودن این روش لگاریتمی بودن آن است.

برای جلوگیری از محاسبه دوباره اعداد می‌توانیم هر عدد را فقط بار اول حساب کرده و در یک دیکشنری ذخیره کنیم و دفعات بعد به جای محاسبه آن، مقدارش را از دیکشنری بخوانیم:

```
let fibo2 n =
  let dict = new System.Collections.Generic.Dictionary<int,int>()
  let rec memoized_fibo n =
    if n = 1 or n = 2 then 1
    else if dict.ContainsKey(n)
    then dict.[n]
    else
      let res = memoized_fibo (n - 1) + memoized_fibo (n - 2)
      dict.[n] <- res
      res
  memoized_fibo n
```

این روش آنقدر سریع است که نتوانستم زمان آن را اندازه بگیرم (خودتان امتحان کنید!)

این تکنیک مختص زبان‌های تابعی نیست و در دیگر زبان‌ها نیز قابل استفاده است.

در درس بعدی با pattern matching و دیگر ساختارهای پیشرفته و متمایز زبان‌های تابعی (و #F) آشنا می‌شویم.

```
[1..50] |> List.filter(fun x -> x %
2 = 0) |> List.map(fun x -> x * x)
|> List.reduce(fun x y -> x + y) |>
printf "%d"
```

این کد هم خوانایی بیشتری دارد و هم مطابق صورت مسئله است؛ در این کد لیست اعداد ۱ تا ۵۰ را به عنوان آرگومان به تابع filter می‌دهیم تا فقط اعداد زوج را انتخاب کند؛ یعنی در این مرحله لیست جدیدی داریم که شامل اعداد زوج بین ۱ تا ۵۰ است؛ سپس این لیست جدید را به عنوان آرگومان به تابع map می‌دهیم تا اعداد را مربع کند. سپس در مرحله بعد اعداد مربع شده را با هم جمع کرده و سپس چاپ می‌کنیم. در واقع ما توسط این عملگر آخرین آرگومان یک تابع را به آن پاس می‌دهیم.

همانطور که قبلاً گفتیم مقادیر در زبان‌های تابعی immutable یا تغییرناپذیر هستند. یعنی ما فقط در هنگام تعریف یک مقدار می‌توانیم عمل انتساب را انجام دهیم.

در برخی از زبان‌های تابعی که تابعی محض هستند (مثل Haskell) این تنها راه است و ما نمی‌توانیم مقداری داشته باشیم که قابل تغییر باشند؛ اما در دیگر زبان‌های تابعی غیرمحض (مثل #F، Scala، OCaml ...) این امکان وجود دارد. به چنین مقداری mutable گفته می‌شود که توسط همین کلمه نیز تعریف می‌شوند. البته عمل انتساب در #F توسط عملگری انجام می‌شود که شاید تا به حال ندیده باشید:

```
let x = 10
let mutable y = 10
y <- 90
```

خط اول یک مقدار غیر قابل تغییر تعریف می‌کند. در خط دوم یک مقدار قابل تغییر تعریف می‌کنیم سپس در خط بعد مقدار آن را توسط عملگر <- تغییر می‌دهیم.

در #F برای انتساب در لحظه تعریف از = استفاده می‌کنیم. در دیگر جاها، عملگر = برای تست مساوی بودن دو مقدار به کار می‌رود (#F عملگر == ندارد)

تکنیک Memoization :

الگوریتم‌هایی وجود دارند که نیازمند محاسبه چندباره یک مقدار هستند. مثلاً الگوریتم محاسبه سری فیبوناچی: با رسم درخت این الگوریتم (در حالت بازگشتی) متوجه خواهید شد که مثلاً در محاسبه دهمین عدد در سری، مقدار عدد پنجم هشت بار محاسبه می‌شود.



تلفن: ۴۶۳ ۷۳۸۴ ۷۷ - ۰۲۱

<http://parmidaco.net>



شرکت پیشرو ارتباط بارمیدا

طراحی و برنامه نویسی تحت وب

صدور گواهینامه های دیجیتال SSL

تجارت الکترونیک و فروش آنلاین



آیا برنامه نویسی خوبی هستید؟

behrouz.rad@gmail.com

آیا برنامه نویسی خوبی هستید؟

برنامه نویسی شغل جالب توجهی است. به عبارتی دیگر، صنعت برنامه نویسی، تنها صنعتی است که وضعیتی عجیب و نامطلوب دارد! این صنعت با سرعت غیر قابل وصفی پیش می رود و شاید بتوان گفت تنها علمی که سرعت تغییرات در آن با علوم مرتبط با برنامه نویسی قابل قیاس است، بیوتکنولوژی است. به هر حال آنچه که واضح است، تاثیر غیر قابل انکار صنعت برنامه نویسی بر دیگر صنایع است.

به هر صنعتی نگاه کنید، مطمئناً بخشی از آن با برنامه نویسی در ارتباط است. از همین رو، اکثر شرکت هایی که در زمینه صنایع مختلف فعالیت می کنند، معمولاً برنامه نویسی یا فردی را که با برنامه نویسی - هر چند اندک - آشنا باشد در اختیار دارند. در ساده ترین حالت، شرکت بخشی با عنوان IT را در زیر مجموعه خود خواهد داشت.

ارگان های مختلف همچون بانک ها، نهادهای آموزشی، ادارات دولتی و ... به برنامه نویسی یا حداقل فردی که با نرم افزارها و جنبه های مختلف کارکرد آنها آشنایی داشته باشد نیازمند هستند. مطمئناً این نیاز، هم برای شما به عنوان یک برنامه نویسی و صد البته برای من نیز! بسیار خوشایند است.

با توجه به بحران اقتصادی که گریبان گیر اقتصاد جهانی شده و اقتصاد ما نیز بی تاثیر از آن نبوده است، آشنایی با فنی که این

بحران کمترین تأثیر را بر روی آن گذاشته و همچنان به عنوان پیر درآمدترین شغل در دنیا محسوب می شود، می تواند هر فردی را برای یادگیری آن وسوسه کند! و آن فن چیزی نیست جز، "برنامه نویسی".

محصول تولید شده توسط یک برنامه نویسی که از آن به عنوان "نرم افزار" یاد می شود، جنبه ی فیزیکی ندارد. در حقیقت، بیشتر هوش، نبوغ و خلاقیت است که در محصول نهایی بروز می کند. نرم افزار، می تواند در چند ثانیه در سرتاسر دنیا پخش شود و اگر مشکلی داشت، در کوتاهترین زمان ممکن مشکل آن برطرف و مجدداً به دست کاربر برسد. آیا محصول دیگری را می شناسید که چنین قابلیت هایی داشته باشد؟

متأسفانه در حال حاضر، برنامه نویسی همانند زمانی است که بشر هنوز به تمدن امروزی خود نرسیده بود و در نتیجه قانون مشخصی برای عملکردش وجود نداشت.

برخی شرکت ها، نرم افزارهایی بسیار عالی و قابل توجه تولید می کنند اما اکثریت به گونه ای عمل می کنند که گویی در حال ایجاد خانه ای کاهگلی هستند! نه مهارتی، نه خلاقیتی، نه کیفیتی و نه اهمیت به رضایت کاربر نهایی! نه کنترلی توسط نهادی معتبر بر روی محصول انجام می گیرد و نه کیفیت آن ارزیابی می شود. تمام این ها موجب ایجاد صنعتی شده است که شبیه به هیچ صنعت دیگری نیست. هیچ صنعتی همچون برنامه نویسی، مهیج، دارای ریسک بالا، دارای



فشاری از خارج برای بهبود عملکرد خود دارند اما واقعیت تلخ تر این است که در حال حاضر چنین اهرمی آن طور که باید و شاید وجود ندارد. پس تنها راه، داشتن انگیزه‌های ذاتی و درونی است و این انگیزه نه قابل یاد دادن است و نه قابل یادگیری.

مشکل دیگر، نبود فشار از جانب مدیر شرکت و در کل، مسئولین پروژه برای تحت فشار قرار دادن برنامه‌نویسان و در اختیار قرار دادن مقداری مشخص از زمان کار فرد در شرکت به منظور آموختن علوم جدید است.

در حال حاضر، اکثر مدیران شرکت‌ها گمان می‌کنند که اختصاص زمان به برنامه‌نویسان در شرکت به منظور یادگیری، کاری نامعقول است در حالی که اثرات این زمانبندی، در کیفیت و سرعت تولیدات نرم افزاری شرکت، در طول زمان به گونه‌ای محسوس قابل مشاهده خواهد بود.

به نظر شما آشنایی هر چه بیشتر با علوم و نرم افزارهایی که محصول شرکت بر پایه‌ی آن‌ها ایجاد می‌شود، معقول نیست؟ اگر به ساختار نرم‌افزارهای قدرتمند و خوش ساخت نگاه کنید، مطمئناً به جدید بودن تکنولوژی‌ها و تکنیک‌های استفاده شده در آن‌ها پی خواهید برد.

حال در نظر بگیرید که یک تکنولوژی در اختیار فردی قرار بگیرد که دانش کافی را برای بکارگیری صحیح آن در اختیار ندارد. به عنوان نمونه می‌توان به تکنولوژی ASP.NET اشاره کرد که برخی به گونه‌ای با آن کار می‌کنند که گویا با ASP کلاسیک روبرو هستند!

البته من خودم را یک برنامه‌نویس نابغه و حرفه‌ای نمی‌دانم. هر چند که در رشته‌ی نرم افزار کامپیوتر تحصیل کرده‌ام، اما اکنون بسیاری از مفاهیمی را که در آن دوران یاد گرفتم فراموش کرده‌ام. از الگوریتم‌هایی همانند Dijkstra و Prim که در درس طراحی الگوریتم می‌خواندیم فقط همین را می‌دانم که برای پیدا کردن کوتاه‌ترین مسیر استفاده می‌شدند و اگر قصد داشته باشم که آن‌ها را پیدا سازی کنم، بدون آن‌ها مجدداً مستندات و نحوه‌ی عملکرد آن‌ها را ببینم، هرگز نمی‌توانم.

البته این را مطمئنم که هرگز نیاز به پیاده‌سازی چنین الگوریتم‌هایی پیدا نمی‌کنم و اگر برنامه‌نویسی نیز با این الگوریتم‌ها آشنایی نداشته باشد، وی را سرزنش نخواهم کرد اما مطمئناً در صورتی که "تمایل" به آشنایی با آن‌ها نداشته باشد او را سرزنش می‌کنم.

وبلاگ‌ها، کتاب‌ها، افرادی که در محیط کار همکار ما هستند، از جمله منابعی می‌باشند که می‌توانند ما را با علوم جدید آشنا کنند؛ به شرطی که ما نیز علاقه مند به یادگیری آن‌ها باشیم.

خلاصه این که باید بدانید که سطح دانش کنونی شما کافی نیست و همیشه در حال یافتن راه‌های نو و یادگیری علوم جدید باشید.

رضایت بسیار از پایان کار و مشکلات فراوان در صورت بروز خطا در کار نیست. هیچ صنعت دیگری نمی‌توان پیدا کرد که شرکت‌های فعال در آن حوزه در کمترین زمان بتوانند خود را بسیار مطرح کنند یا بالعکس، نابود شوند! آیا صنعت دیگری را می‌شناسید که چنین قابلیت‌هایی داشته باشد؟

و اما...

تمام فاکتورهایی که ذکر شد، ما را به سمتی سوق می‌دهند که با مشکلی که امروزه با آن مواجه هستیم روبرو کرده است: تقاضای زیاد برای استخدام برنامه‌نویس از سویی و ناپختگی و نداشتن تجربه‌ی کافی برنامه‌نویسان جوان از سویی دیگر!... در این حالت، گروهی از افراد به وجود می‌آیند که متأسفانه با عدم داشتن اطلاعات و تجربه‌ی کافی، صنعت برنامه‌نویسی را با مشکل مواجه می‌کنند.

فکر می‌کنید چه تعداد از برنامه‌نویسان با نحوه‌ی عملکرد کامپیوتر آشنایی دارند؟ یا چند درصد از آن‌ها در مورد نحوه‌ی کار یک کامپایلر اطلاع دارند؟ یا چه تعداد از ساختمان‌های داده‌ی مختلف استفاده می‌کنند بدون آن که در مورد زمان مناسب استفاده از آن‌ها اطلاعی داشته باشند؟

ممکن است بگویید: یک برنامه‌نویس نیازی به دانستن این موارد ندارد. بسیار خوب، اجازه بدهید با شما موافقت کنم اما مشکل این نیست که یک برنامه‌نویس این موارد را نداند، مشکل این است که "او نمی‌خواهد این موارد را یاد بگیرد!"

این که فردی باهوش، دارای خلاقیت و نبوغ بالا باشد، برای این که وی را یک برنامه‌نویس خوب بدانیم کافی نیست. البته این خصوصیات از جمله خصوصیات لازم برای یک برنامه‌نویس خوب هستند اما مهمترین عامل، "تمایل به یادگیری" است. و این معضلی است که امروزه گریبانگیر صنعت برنامه‌نویسی است.

تثبیت سطح دانش افراد در سطحی مشخص که افزایش آن بسیار کند صورت می‌گیرد از جمله مشکلاتی است که توسعه گران نرم‌افزار با آن مواجه هستند.

برنامه‌نویسان همیشه از کار زیاد و دستمزد کم خود گلایه دارند. اما باید به این پرسش پاسخ داد که فردی که به ازای هر ساعت برنامه‌نویسی - فرضاً با محیط NET - ، به طور متوسط ۳۰۰۰ تومان دستمزد می‌گیرد اما نمی‌تواند تفاوت interface و کلاسی از نوع abstract را توضیح دهد آیا سزاوار توجه بیشتری است؟ من به شخصه معتقدم که ساعتی ۱۰۰۰ تومان نیز برای فردی که تفاوت عضوی از نوع private و protected را نمی‌داند زیاد است!

به نظر من مشکل از اینجا ناشی می‌شود که برنامه‌نویسان و در کل، توسعه‌گران نرم افزار، انگیزه‌ای برای یادگیری و در نتیجه بهتر انجام دادن کار خود ندارند. نداشتن انگیزه از آنجا ناشی می‌شود که فشاری بر روی برنامه‌نویس به منظور ارتقاء سطح دانش خود و ارائه‌ی کار بهتر وجود ندارد.

متأسفانه باید اقرار کرد که برخی از برنامه‌نویسان نیاز به وجود

Win32 در برابر MFC - قسمت اول

mehdi_mousavi@hotmail.com



• Program Initialization (راه اندازی اولیه برنامه)

• Message Loop (چرخه پیام)

هنگامی که اجرای برنامه آغاز می‌شود، ویندوز اطلاعاتی به برنامه پاس می‌کند که از آن جمله می‌توان به Handle نمونه فعلی و قبلی (در صورت وجود) برنامه اشاره کرد. اجازه دهید تا نگاهی به یکی از توابع Win32 که توسط MSVC++ ایجاد شده است، داشته باشیم:

```
int APIENTRY WinMain(HINSTANCE hInstance,
                    HINSTANCE hPrevInstance,
                    LPSTR lpCmdLine,
                    int nCmdShow)
```

اولین پارامتر، hInstance، به صورت int تعریف شده است و بیانگر هندل به نمونه فعلی در حال اجرای برنامه است. پارامتر دوم، hPrevInstance، هندل به نمونه قبلی اجرایی همان برنامه است. البته در API های 32 بیتی ویندوز NT، این پارامتر همواره بدون توجه به نمونه قبلی، NULL است. دلیل این مطلب، در پشت این حقیقت نهفته شده است که هر نسخه از هر برنامه در فضای آدرسی خودش اجرا می‌شود و هیچ یک از منابعش را بنابه دلایل امنیتی، با نمونه‌های دیگر در حال اجرا، به اشتراک نمی‌گذارد.

پارامتر سوم، lpCmdLine، حاوی دستورات خط فرمان برنامه است.

پارامتر آخر، nCmdShow، برای تعیین چگونگی نمایش پنجره، هنگامی که برنامه برای بار نخست اجرا می‌گردد، استفاده می‌شود.

مقدمه: در یک هوای آفتابی در روز ۲۹ می ۲۰۰۱، من و یکی از دوستانم در مورد فوائد MFC و این‌که چگونه زندگی ما را آسان کرده است، در حال گفتگو بودیم. او با مطرح کردن این سوال که "معماری document/view چیست؟"، بحث را آغاز کرد. آنگاه درباره پاسخ این سوال شروع به تفکر کردیم و به زودی دریافتیم که نمی‌دانیم چیست و چگونه MFC در درون خود API های Win32 را جای داده است. هنگامی که سوال دوم مطرح شد، ما هر دو توافق کردیم تا نگاهی به برخی کتب و مقالات بیندازیم و راهی را با برخی افراد ترتیب دهیم تا به پاسخ سوالاتمان برسیم. سوال دوم این بود: "تابع WinMain در کجای یک برنامه MFC قرار گرفته است؟"

این سری مقالات برای رفع عطش آن دسته از افرادی است که از نبود منابع کافی در مورد معماری document/view در MFC رنج می‌برند و عاشقانه آن را به والدینم ص. موسوی و ا. شاهیده تقدیم می‌کنم.

الفبای برنامه‌های Win32

هر برنامه در محیط ویندوز حداقل حاوی دو تابع است: WinMain و wndProc. هر برنامه در محیط ویندوز به تابع WinMain نیاز دارد، زیرا نقطه شروع یک برنامه است. تابع دیگر Window Procedure است که پیام‌های پنجره‌ای را پردازش می‌کند (اگرچه این اسم بسیار دقیق نیست، زیرا این تابع کلیه پیام‌های ارسالی به برنامه را پردازش می‌کند).

اجازه دهید تا ببینیم WinMain چیست و در مورد وظایفش صحبت کنیم. سپس دنده را برای رسیدن به wndProc عوض خواهیم کرد.

تابع WinMain به سه بخش تقسیم می‌شود:

• Procedure Declaration (بیان رویه)

هنگامی که این کار را انجام می‌دهیم، ویندوز عناصر ساختار را در محلی به نام Class Database کپی می‌کند. هنگامی که برنامه‌ای می‌خواهد پنجره‌ای ایجاد کند، برنامه به مدخلی در Class Database مراجعه کرده و سپس ویندوز از اطلاعات مزبور در این مدخل، برای ایجاد پنجره استفاده می‌کند. جالب است، این طور نیست؟

اکنون زمان بررسی اعضای این ساختار رسیده است اما از آنجایی که تعداد اعضای آن زیاد است، این وظیفه را به دوش خواننده می‌گذاریم تا کاربرد هر یک از اعضا را کشف کنند. با این وجود، ما به یکی از اعضای این ساختار علاقه‌مندیم، زیرا ما را به سوی Window Procedure - دومین تابع برنامه - رهنمون می‌سازد.

بله، این عضو lpfnWndProc است. اما دست نگه دارید! ابتدا باید بخش آغازین برنامه را در تابع WinMain تکمیل کنیم. پس از ثبت داده‌های کلاس، تابع CreateWindow را به منظور ایجاد پنجره فراخوانی می‌کنیم:

```
HWND hWnd = CreateWindow("myClass",
    "WindowTitle",
    WS_OVERLAPPEDWINDOW,
    CW_USEDEFAULT,
    CW_USEDEFAULT,
    CW_USEDEFAULT,
    CW_USEDEFAULT,
    NULL,
    NULL,
    hInstance,
    NULL);
```

و سپس پنجره را با استفاده از تابع ShowWindow نمایش می‌دهیم:

```
ShowWindow(hWnd, nCmdShow);
```

این تمام مطلبی است که "راه اندازی اولیه برنامه" در موردش صحبت می‌کند. اکنون به عقب بازگشته و به سراغ عضو lpfnWndProc از ساختار WNDCLASSEX می‌رویم. همانطور که خودتان هم از نام مجارستانی این عضو پی برده‌اید، این عضو اشاره‌گری به تابعی بنام Window Procedure است. ما این عضو را به wndProc نسبت داده‌ایم، بنابراین تابعی با نام wndProc در برنامه خودمان تعریف می‌کنیم که وظیفه‌اش، پردازش پیام‌های ارسالی به پنجره است. اکنون، زمان آن رسیده است که دنده را از "راه‌اندازی اولیه برنامه" به "چرخه پیام‌ها" تغییر دهیم، سومین و آخرین قسمت از تابع WinMain.

این پارامتر به ویندوز می‌گوید که پنجره اصلی برنامه ما را چگونه نمایش دهد، Maximize، Minimize و ... Prototype مقدار بازگشتی تابع اخیر، APIENTRY می‌باشد، که به صورت زیر تعریف شده است:

```
#define APIENTRY WINAPI
```

و به همین ترتیب، WINAPI بصورت زیر:

```
#define WINAPI __stdcall
```

و __stdcall چیست؟ متأسفانه پاسخ به این سوال از حیطه این مقاله خارج است، بنابراین یافتن پاسخ این سؤال را به خوانندگان می‌سپاریم. در هر حال، تمام چیزی که "بیان رویه" درباره آن صحبت می‌کند، همین است.

اکنون، نوبت فاز دوم است: "راه‌اندازی اولیه برنامه". این فاز، شامل فراخوانی سه رویه ویندوز است:

- RegisterClass یا نسخه بسط یافته آن RegisterClassEx.
- CreateWindow یا نسخه متناظر بسط یافته‌اش، CreateWindowEx.
- ShowWindow (این رویه، نسخه بسط یافته ندارد!)

برای ایجاد یک پنجره، باید اعضای ساختار WNDCLASSEX را پر کرده و سپس نمونه‌ای از این ساختار را به تابع RegisterClassEx پاس دهیم. روش کار بدین ترتیب است:

```
WNDCLASSEX wcl;
wcl.cbSize = sizeof(WNDCLASSEX);
wcl.hInstance = hInstance;
wcl.lpfnWndProc =
    (WNDPROC)wndProc;
wcl.style = CS_HREDRAW | CS_VREDRAW;
wcl.hIcon = LoadIcon(hInstance,
    IDC_ARROW);
wcl.hIconSm = NULL;
wcl.hCursor = LoadCursor(NULL,
    IDC_ARROW);
wcl.lpszMenuName = NULL;
wcl.cbClsExtra = 0;
wcl.cbWndExtra = 0;
wcl.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH);
wcl.lpszClassName = "myClass";

if(!RegisterClassEx(&wcl))
    return 0;
```

که در آن، LRESULT به صورت نوع داده‌ای long و CALLBACK سبک فراخوانی __stdcall را تعیین می‌کند. هرگاه که پیامی تولید می‌شود، Window Procedure، فراخوانی می‌شود. در صورت نیاز می‌توانیم پیام را بررسی کنیم، در غیر این صورت، آن را به تابع DefWindowProc پاس خواهیم داد. تابع DefWindowProc هر کاری که برای کارکرد صحیح جعبه پنجره ما نیاز است، انجام خواهد داد. شرکت مایکروسافت، Source این تابع را از طریق Windows SDK در اختیار ما قرار داده‌است. به هر حال، به محض دریافت پیام WM_DESTROY، تابع PostQuitMessage را فرا می‌خوانیم، که امور مربوط به خاتمه برنامه را انجام می‌دهد.

الفبای برنامه‌های MFC

یک برنامه MFC توابع Win32 فوق‌الذکر را به گونه‌ای کپسوله کرده‌است که زندگی هر برنامه‌نویسی آسان‌تر شود. ما همگی حتماً درباره نقطه شروع یک برنامه MFC در کتب و مقالات متفاوت مطلب خوانده‌ایم. همه آن‌ها کم و بیش ذکر کرده‌اند که مدخل ورودی یک برنامه MFC، متود InitInstance است. این مطلب، سؤال جدیدی را به وجود می‌آورد: "پس تابع WinMain چه می‌شود، اگر InitInstance تابع ورودی برنامه باشد؟" برای روشن شدن آن چه که پشت پرده رخ می‌دهد، مایلیم تا برنامه‌ای نمونه برای روشن شدن جزییاتی که از دید یک برنامه نویس MFC پنهان مانده‌است، ایجاد کنم. برای ایجاد این برنامه، MSVC++ 6.0 را اجرا کنید. از منوی فایل، گزینه new item را انتخاب کنید. در حالی که tab پروژه انتخاب شده‌است، گزینه (MFC App Wizard) را انتخاب کرده و در محلی که برای نام پروژه در نظر گرفته شده‌است، کلمه sdisample را بنویسید و کلید OK را فشار دهید. گزینه Single Document را انتخاب کرده، کلید Finish را فشار دهید. کلید OK را فشار دهید تا Wizard برنامه مورد نظر شما را ایجاد کند.

در نگاه اول، درخواهید یافت که برنامه حاوی کلاس‌های زیر است:

- CAboutDlg
- CMainFrame
- CSdiSampleApp
- CSdiSampleDoc
- CSdiSampleView

و یکی از این کلاس‌ها با نام CSdiSampleApp از کلاس CWinApp مشتق شده است:

هر برنامه‌ای در ویندوز، حاوی یک چرخه پیام (Message Loop) است تا به برنامه اجازه بررسی پیام‌ها را بدهد. بدین ترتیب هر برنامه با تدارکات پیام‌ها در ارتباط خواهد بود که این امر، لزوم کارکرد صحیح یک برنامه در سیستم عامل ویندوز است. اینجا یک نمونه عمومی از چرخه پیام را می‌بینید:

```
MSG msg;
while (GetMessage(&msg, NULL, 0, 0))
{
    TranslateMessage(&msg);
    DispatchMessage(&msg);
}
```

این چرخه به طور مداوم کار می‌کند تا پیام WM_QUIT دریافت شود. به محض وصول این پیام، برنامه چرخه پیام را شکسته و به کار خود خاتمه می‌دهد. واضح است که هر گذر در حلقه فوق به معنای دریافت یک پیام است، خواه این پیام از طریق "صف رخدادهای سخت افزار" باشد، خواه از طریق "صف پیام‌های برنامه". همانطور که متوجه شده‌اید، چرخه پیام‌ها شامل سه رویه اصلی است: GetMessage که پیام را به برنامه ما می‌آورد، TranslateMessage که پیام‌های مربوط به ضرب کلیدها را به مقادیر معتبر کاراکتری ترجمه کرده و در صف پیام‌های خصوصی برنامه تحت قالب پیام WM_CHAR قرار می‌دهد و آخرین رویه، DispatchMessage، که پیام بازبایی شده را (msg) به منظور پردازش، به سمت Window Procedure برنامه گسیل می‌کند. با در دست داشتن این اطلاعات، اکنون آماده‌ایم تا کارکرد یک Window Procedure را دریابیم. به خاطر می‌آورید که عضو lpfnWndProc از ساختار WNDCLASSEX را به wndProc نسبت دادیم. اکنون ببینیم که wndProc به چه شباهت دارد:

```
LRESULT CALLBACK wndProc(HWND hWnd,
UINT message, WPARAM wParam, LPARAM lParam)
{
    switch (message)
    {
        case WM_DESTROY:
            PostQuitMessage(0);
            break;
        default:
            return
            DefWindowProc(hWnd, message, wParam, lParam);
    }
    return 0;
}
```

و `tWinMain_` چطور؟ `tWinMain_` نیز به صورت زیر تعریف شده است:

```
#define _tWinMain WinMain
```

شگفتا! باز هم به نقطه قبلی رسیدیم. تابع `WinMain` ای که در ابتدا درباره‌اش سخن گفتیم، بدین صورت است:

```
int APIENTRY WinMain(HINSTANCE hInstance,
    HINSTANCE hPrevInstance,
    LPSTR lpCmdLine,
    int nCmdShow)
```

و این تابع تولید شده برنامه MFC است:

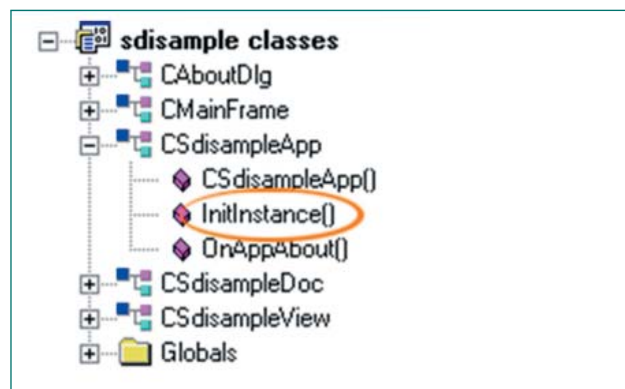
```
extern "C" int WINAPI _
tWinMain(HINSTANCE hInstance,
    HINSTANCE hPrevInstance,
    LPTSTR lpCmdLine,
    int nCmdShow)
```

این دو تابع را با هم مقایسه کنید و خواهید دید هر دو تابع یکی هستند. جالب است، نه؟ اکنون اجازه دهید نگاه دقیق‌تری به `tWinMain_` و نحوه پیاده‌سازی آن داشته باشیم:

```
extern "C" int WINAPI _
tWinMain(HINSTANCE hInstance,
    HINSTANCE hPrevInstance,
    LPTSTR lpCmdLine,
    int nCmdShow)
{
    // call shared/exported WinMain

    return AfxWinMain(hInstance,
        hPrevInstance, lpCmdLine, nCmdShow);
}
```

همان طور که می‌بینید، این رویه، تابع `AfxWinMain` را فراخوانی می‌کند؛ تابع `WinMain` برنامه‌های MFC. اما آن چیست و چگونه پیاده‌سازی شده است؟ قسمت‌های مهم این تابع در زیر نمایش داده شده است:



کلاس فوق‌الذکر حاوی تابعی با نام `InitInstance` است، که ادعا شده است که نقطه شروع یک برنامه MFC است. سؤال آن است که چرا این تابع به عنوان مدخل ورودی تعریف شده است؟ پاسخ به این سؤال، پشت معماری MFC قرار گرفته است. گنگ است، این طور نیست؟

قبلاً گفتیم که هر برنامه‌ای در Windows حاوی دو تابع است: `WinMain` و `wndProc`. اکنون می‌گوییم که همین مطلب درباره برنامه‌های MFC نیز صادق است. برنامه‌های MFC نیز، حاوی دو تابع `WinMain` و `wndProc` هستند. برنامه را با فشار کلید F10 اجرا کنید. خواهید دید که اجرای برنامه در کد زیر متوقف خواهد شد:

```
extern "C" int WINAPI _
tWinMain(HINSTANCE hInstance,
    HINSTANCE hPrevInstance,
    LPTSTR lpCmdLine,
    int nCmdShow)
{
    // call shared/exported WinMain

    return AfxWinMain(hInstance,
        hPrevInstance, lpCmdLine, nCmdShow);
}
```

به دقت نگاه کنید! این تابع، همان پارامترهای آشنای `WinMain` را داراست. اما آن عبارات زشت قبل از `WINAPI` چیست؟ عبارت `extern "C"` به کامپایلر می‌گوید که چگونه این تابع را کامپایل کند و `WINAPI` نیز به صورت زیر تعریف شده است:

```
#define WINAPI __stdcall
```

همان طور که می‌بینید، ابتدا تابع `AfxWinInit` فراخوانی شده‌است، تابعی که باعث بارگذاری `MFCO42D.DLL` (اگر نسخه `Debug` برنامه اجرا شود)، تنظیم نام فایل اجرایی، فایل `Help` و فایل `Ini` برنامه می‌شود (اگرچه این تابع وظایف دیگری نیز انجام می‌دهد، اما قرار نیست ما `MFC` را در این مقاله دوباره طراحی کنیم، بنابراین آن‌ها را نادیده می‌گیریم). سپس تابع `InitApplication` فراخوانی می‌شود، تابعی که به راحتی می‌توانیم در کلاس `CSdiSampleApp` آنرا `Override` کنیم. این تابع برای مقداردهی و راه‌اندازی یکباره برنامه مورد استفاده قرار می‌گیرد. سپس، چند خط بعد، تابع `InitInstance` فراخوانی می‌شود، تابعی که ادعا کردیم نقطه ورودی یک برنامه `MFC` است. سپس تابع `AfxWinMain`، تابع `Run` را فراخوانی می‌کند. این تابع نیز به نوبه خود، متد `CWinApp::Run` را فراخوانی می‌کند و تابع اخیر نیز متد `CWinThread::Run` را فراخوانی می‌کند. چرخه پیام‌ها در این متد قرار گرفته و به صورت زیر پیاده‌سازی شده‌است:

```
int CWinThread::Run()
{
    ASSERT_VALID(this);

    // for tracking the idle time
    state

    BOOL bIdle = TRUE;
    LONG lIdleCount = 0;

    // acquire and dispatch messages
    until

    // a WM_QUIT message is re-
    ceived.

    for (;;)
    {
        // phase1: check to see if
        we can do idle work

        while (bIdle && !::PeekMes-
        sage
            (&m_msgCur, NULL, NULL,
            NULL, PM_NOREMOVE))
        {
            // call OnIdle while in
            bIdle state
```

```
int AFXAPI AfxWinMain(HINSTANCE hIn-
stance,
                    HINSTANCE
hPrevInstance,
                    LPTSTR lpCmd-
Line,
                    int nCmdShow)
{
    //Snipped

    // AFX internal initialization

    if (!AfxWinInit(hInstance, hPre-
vInstance, lpCmdLine, nCmdShow))
        goto InitFailure;

    // App global initializations
    (rare)

    if (pApp != NULL && !pApp-
>InitApplication())
        goto InitFailure;

    // Perform specific initializa-
    tions

    if (!pThread->InitInstance())
    {
        if (pThread->m_pMainWnd !=
        NULL)
        {
            TRACE0("Warning: De-
stroying non-NULL m_pMainWnd\n");
            pThread->m_pMainWnd-
>DestroyWindow();
        }
        nReturnCode = pThread-
>ExitInstance();
        goto InitFailure;
    }

    nReturnCode = pThread->Run();

    //Snipped

}
```



```
MSG msg;
while (GetMessage(&msg, NULL, 0,
0))
{
    TranslateMessage(&msg);
    DispatchMessage(&msg);
}
```

با این حال، تفاوت‌هایی بین این دو چرخه وجود دارد. اولین و مهم‌ترین این تفاوت‌ها، آن است که `CWinThread::Run` به گونه‌ای طراحی شده که تابع `OnIdle` را هنگامی که پیامی در صف وجود ندارد، فراخوانی کند. ثانیاً، این تابع پس از دریافت پیام `WM_QUIT` و قبل از خروج از برنامه، تابع `ExitInstance` را فراخوانی می‌کند. بنابراین برنامه‌نویس MFC می‌تواند با Override کردن تابع `ExitInstance`، کلیه امور مورد نظرش را قبل از خاتمه برنامه انجام دهد.

از طرف دیگر، `PumpMessage`، توابع `TranslateMessage` و `DispatchMessage` را در درون خود فراخوانی می‌کند. اکنون می‌بینیم که چرخه پیام‌های برنامه‌های MFC نیز در جایگاه خود قرار دارد.

با این حال ما در مورد `wndProc` سخن نگفتیم. این تابع، از نظر من، زیباترین بخش MFC است که در قسمت دوم این مقاله به توضیح آن خواهیم پرداخت، با این امید که این سری مقالات تحت عنوان "MFC در برابر Win32"، درهای جدیدی به روی شما باز کند.

```
if
(!OnIdle(lIdleCount++))
    bIdle = FALSE; //
assume "no idle" state

}

// phase2: pump messages
while available

do
{
    // pump message, but
quit on WM_QUIT

    if (!PumpMessage())
        return ExitIn-
stance();

    // reset "no idle"
state after

    // pumping "normal"
message

    if (IsIdleMessage(&m_
msgCur))
    {
        bIdle = TRUE;
        lIdleCount = 0;
    }
} while (::PeekMessage(&m_
msgCur,
NULL, NULL, NULL, PM_
NOREMOVE));

ASSERT(FALSE); // not reach-
able

}
```

همانطور که مشاهده می‌کنید، این تابع حاوی یک چرخه است و این چرخه در صورت دریافت پیام `WM_QUIT` شکسته خواهد شد و بنابراین همان وظیفه چرخه پیام‌ها که در ابتدا مورد بررسی قرار گرفت را دارا می‌باشد:

Anders Hejlsberg

آشنایی با بزرگان

نویسنده: مهدی عسگری

مایکروسافت داشت که یکی از بالاترین سطوح کار در این شرکت است. وی از سال ۲۰۰۰ تا الان بر روی طراحی و پیاده‌سازی زبان C# کار می‌کند که در سال ۲۰۰۱ جایزه Dr. Dobb's Excellence in Programming Award را به خاطر این زبان دریافت کرد (این جایزه هر ساله به کسی داده می‌شود که تأثیر زیادی به روی پیشرفت نرم‌افزار داشته است. قبلاً به وجود آورندگان زبان‌های پایتون، پرل و جاوا نیز به خاطر آن زبان‌ها این جایزه را دریافت کرده‌اند). وی از ابتدای ایجاد .NET Framework جزئی از معماران آن بوده است (در واقع طراحی دات و نت و سی شارپ هر دو از هم الهام گرفته‌اند)

وی بعدها به Technical Fellow ارتقاء یافت که بالاترین سطح یک کارمند در مایکروسافت می‌باشد (در حال حاضر حدود ۱۹ نفر در کل شرکت مایکروسافت TF هستند)

فایل‌های ویدئویی ارائه‌های ایشان در کنفرانس‌های Lang.Net ، JAOO ، PDC ، و ... به رایگان در Channel9 مایکروسافت موجود و قابل مشاهده/دانلود می‌باشند. همواره محور صحبت‌های ایشان حول محور C# و پاسخ به ابهام‌ها و صحبت درباره‌ی آینده‌ی این زبان می‌چرخد. (گاهی نیز با خالق دیگر زبان‌ها مصاحبه کرده است مثل Guy Steele ، Gilad Bracha و ...)

از ایشان یک کتاب به نام The C# Programming Language چاپ شده (انتشارات Addison-Wesley) که به نوعی استاندارد و specification زبان محسوب می‌شود.

برای آشنایی بیشتر با این شخص فیلمی در بخش Behind The Code کانال ۹ مایکروسافت وجود دارد که محور مصاحبه این دفعه شخص ایشان است نه موضوعات مربوط به فن‌آوری‌هایی مثل دات نت یا زبان سی شارپ که از این آدرس قابل دانلود است:

<http://channel9.msdn.com/shows/Behind+The+Code/Life-and-Times-of-Anders-Hejlsberg/>



در هر شماره در این ستون به معرفی افراد بزرگی که در تاریخ برنامه‌نویسی تأثیر بزرگی داشته‌اند می‌پردازم. آقای Hejlsberg در سال ۱۹۶۰ در کپنهاگن (پایتخت دانمارک) متولد شدند. ایشان در دانشگاه فنی دانمارک به تحصیل پرداختند و در اوایل دهه ۸۰ به نوشتن برنامه‌هایی برای یکی از شرکت‌های دانمارکی مشغول شدند که یکی از آن برنامه‌ها کامپایلری برای زبان پاسکال بود. با فروختن لایسنس کامپایلرش به شرکت Borland (و بعدها استخدام در آن شرکت و مهاجرت به آمریکا) برگ جدیدی از تاریخ برنامه‌نویسی ورق خورد؛ ایشان در بورلند کامپایلر و محیط توسعه‌ی Turbo Pascal را فروختند که بنا بر ادعای خیلی‌ها بهترین (و پر فروش‌ترین) کامپایلر پاسکال تاریخ شد (کسانی که آن موقع برنامه‌نویس بودند به یاد دارند که TP برنامه‌ها را سریع‌تر از دیگر کامپایلرها کامپایل می‌کرد؛ این سرعت + IDE خوب + قیمت نسبتاً پایین آن (۵۰ دلار) باعث موفقیت این محصول برای شرکت و شخص Hejlsberg شد). پس از TP آقای Hejlsberg روی زبان Delphi کار کرد (به عنوان معمار اصلی این برنامه) در سال ۱۹۹۶ ایشان در پی مخالفت‌هایی با مدیریت بورلند از آن در آمده و در مایکروسافت مشغول به کار شدند. اولین محصول ایشان در مایکروسافت، زبان J++ و بعد Windows Foundation، و در Classes بود. وی عنوان Distinguished Engineer را در

نویسنده: حسین جزایری

روبا دوست داشتنی با امکاناتی جدید

h.jaza@yahoo.com



حذف آن‌ها)، History و اساساً کلیه تغییرات صورت گرفته در زمان مرور وب سایت‌ها را به ما می‌دهد.

• Web Workers :

این ویژگی به مرورگر این امکان را می‌دهد که وظایف محاسباتی را بتوان در پشت صحنه انجام داد که پتانسیل استفاده از برنامه‌های پیچیده را بالاتر می‌برد. در واقع Web Workers قابلیت است که این اختیار را به مرورگر می‌دهد که کدهای جاوا اسکریپتی که نیاز به پردازش سنگینی دارند را در یک Thread جداگانه و مجزا از Thread مربوط به لود صفحه مربوطه، اجرا کند.

• کارآیی و پایداری بیشتر به کمک موتور جاوا اسکریپت TraceMonkey :

سرعت لود پایین JavaScript، یکی از مشکلات بزرگی است که موزیلا همواره با آن مواجه بوده اما این بار، با بهره‌گیری از موتور جاوا اسکریپت TraceMonkey، که به صورت پیش فرض نیز فعال است، راهکار مناسبی ارائه شده است. این موتور جدید نسبت به سایر انجین‌های جاوا اسکریپت، سرعت بالاتری دارد و بر اساس آمارهای منتشر شده، ۱،۲ TraceMonkey بار سریع‌تر از موتور ۷۸ کروم و بین ۵ تا ۲۰ برابر هم سریع‌تر از سایر موتور جاوا اسکریپت مرورگرهای دیگر می‌باشد؛ در نتیجه با بهره‌گیری از این موتور، جاوا اسکریپت‌های موجود در صفحات سریع تر لود خواهند شد.

• بهره‌گیری از JSON :

JSON یا همان JavaScript Object Notation، امکان تبادل اطلاعات بین مرورگر و سرور را فراهم می‌سازد. مسئله‌ای که موزیلا به سادگی طراحی آن، تأکید داشته و افزایش سرعت و امنیت را از خصوصیات آن ذکر می‌کند. برای مثال، اگر بخواهیم سازگاری را هم برای فایر فاکس ۳ و هم ۳،۵ تضمین کنیم، می‌توان به این گونه عمل کرد:

پروژه‌ی فایر فاکس در سال ۲۰۰۳ توسط Dave Hyatt و Blake Ross در شرکت موزیلا، آغاز شده و یک سال بعد، در نوامبر ۲۰۰۴ اولین نسخه‌ی آن عرضه شد. ویژگی‌های این محصول از جمله cross-platform بودن و بهره‌گیری از XUL که پتانسیل اضافه شدن افزونه‌ها و استفاده‌ی تم‌ها را برای آن امکان پذیر می‌کرد، باعث شد تا از همان ابتدا با مقبولیت عمومی مواجه شود. ادامه‌ی این پروژه، افزوده شدن هر چه بیشتر امکاناتی از جمله tabbed-browsing، spell checker، incremental live bookmarking، find، download manager و رفع مشکلات و نواقص قبلی، محبوبیت این مرورگر را تا جایی بالا برد که طبق آمار منتشر شده در ژوئن امسال ۲۰۰۹، بیست و دو درصد از استفاده‌ی کاربران را به خود اختصاص داده و توانسته از رقبای قدرتمند خود مثل Safari و Opera پیشی بگیرد و بعد از Internet Explorer، به عنوان دومین مرورگر محبوب و پرکاربرد شناخته شود. این مقبولیت عام، حتی در بعضی موارد، ماکروسافت را نیز مجبور به تغییر سیاست‌هایی در استانداردها و به تبع، مرورگرش کرد یا در واقع استانداردهای ساختگی و غیر اصولی مرورگر ماکروسافت را توانست زیر سوال ببرد و وی را به تبعیت و رعایت آن اصول اصلی سوق دهد.

فایر فاکس ۳،۵ که شیرتوکو^۱ نام گرفته، به عنوان نسخه‌ی در حال توسعه‌ی کنونی دستخوش تغییرات زیادی شده. لازم به ذکر است که قبلاً این نسخه به عنوان فایر فاکس ۳،۱ شناخته می‌شد لکن این تغییرات زیاد و البته گسترده، مسئولان موزیلا را قانع کرد تا آن را با نگارش ۳،۵ برای عرضه آماده کنند. تغییرات عمده‌ی این نگارش عبارتند از:

• افزوده شدن حالت Private Browsing :

Private Browsing یکی از امکاناتی است که توسط Chrome و Safari عرضه شد و مزایای بهره‌گیری از آن، موجب شد تا فایرفاکس هم آن را به نسخه‌ی جدید مرورگر خود، بیفزاید. این امکان، قابلیت عدم ذخیره‌ی Cookie ها (در اصل ذخیره و بعد

می‌شود.

word-wrap: پشتیبانی از این خصوصیت جدید، باعث می‌شود تا از بروز مشکلات **overflow** برای رشته‌های بسیار طولانی، جلوگیری به عمل آید و خصوصیات مشابه دیگری از قبیل - **moz-box-shadow**، - **moz-border-image**، - **moz-column-rule**، - **moz-column-rule-width**، - **moz-column-rule-style**، - **moz-column-rule-color**، - **moz-column-gap** و استفاده از مبدل های CSS مانند **nth-child**، **nth-last-child**، **nth-of-type**، **nth-last-only-of-type**، **first-of-type**، **last-of-type** و **of-type** که همگی در فایر فاکس ۳٫۵ پشتیبانی می‌شوند.

• ویژگی‌های قابل توجهی در مورد DOM:

DOM worker، خصوصیتی است که امکان مالتی ترد کار کردن برنامه‌های وب را به سادگی برای ما فراهم می‌سازد که از جمله‌ی آن به **Web Workers** در بالا اشاره شد. API موقعیتیابی **W3C^۳** که **geolocation** نام دارد و امکان اطلاع از موقعیت جغرافیایی کاربر را برای ما فراهم می‌کند. موقعیتیابی عناصری (از نوع **DOM**) که از گزینشگرها استفاده کرده‌اند.

شیء **NodeIterator** که امکان گردش بین نودهای زیر درخت یک **DOM** را برای ما فراهم می‌سازد. و یکپارچگی **JSON** با **DOM** که همگی از مؤلفه‌های اضافه شده به **DOM** می‌باشند.

• **امکاناتی در زمینه‌ی JavaScript**: از جمله افزوده شدن دو متد **Object.getPrototypeOf** برای دسترسی به **prototype** یک شیء خاص و متد های **Trim** برای سادگی بیشتر کار با رشته‌ها **trim**، **trimLeft** و **trimRight**.

• **افزونه‌ها**: افزونه‌ها نیز که یکی از دلایل اساسی محبوبیت این مرورگر می‌باشند، با تغییراتی باعث هر چه ساده‌تر کار کردن با آنها شده‌اند. از نسخه‌های اولیه تا کنون، دسترسی مستقیم به دیتابیس **Places** با استفاده از **Storage API** صورت می‌گرفت که نیاز به کمی دقت و البته ریز کاری داشت به نحوی که به عنوان مثال به این گونه عمل می‌شد:

```
var places = Components.
classes["@mozilla.org/file/directory_
service;1"].
getService(Components.inter-
faces.nsIProperties).
get("ProfD", Components.inter-
faces.nsIFile);
```

```
if (typeof(JSON) == "undefined")
{
    Components.utils.
import("resource://gre/modules/JSON.
jsm");
    JSON.parse = JSON.fromString;
    JSON.stringify = JSON.to-
String;
}
```

بدین وسیله اگر مدل جاوا اسکریپت **JSON.jsm** اساساً پشتیبانی نشود، با **import** کردن آن این امر صورت می‌پذیرد و سپس با تطبیق دادن متدهایی که از **JSON** پشتیبانی می‌کنند با آن‌هایی که نمی‌کنند، در نهایت به یک فراخوانی مشترک دست خواهیم یافت.

• **ارتقاء و بهبود موتور Gecko (نسخه‌ی 1.9.1) که باعث نمایش سریع تر صفحات شده.**

• افزوده شدن امکاناتی از HTML 5:

یکی از مهم ترین و کارآمد ترین ویژگی‌های این نگارش، پشتیبانی از تگ‌های **video** و **audio** است. با این امکان، دیگر نیازی به نصب فلش پلیر و یا تکنولوژی‌های مشابه، برای پخش فایل‌های صوتی و تصویری نیست. خصوصیتی که در نسخه‌های کنونی، صرفاً محدود به فایل‌های **ogg** می‌شود.

از دیگر ویژگی‌های **HTML 5** که توسط فایر فاکس ۳٫۵ پشتیبانی می‌شود، می‌توان به **Drag and Drop** اشاره کرد. استفاده از این API، باعث می‌شود تا بتوانیم آیتم‌های متفاوت را مابین صفحات وب، جابجا کنیم.

• خصوصیات و ویژگی‌های جدیدی از CSS 2.1 و 3:

پشتیبانی از فونت‌های قابل دانلود^۷: مرورگر می‌تواند صفحه‌ی وب را دقیقاً مطابق با فونت درخواستی نویسنده به نمایش بگذارد. این ویژگی توسط خصوصیت جدید **font-face@** امکان پذیر شده. همچنین واسط‌های جستجویی **CSS^۸** که سطح پشتیبانی از واسط‌های وابسته‌ی مجموعه تعاریف^۹ را ارتقاء می‌دهد: **before** و **after**: شبه عناصری که در **CSS 2.1** آپدیت شده و از **list**، **float**، **position**، **style** * و بعضی خصوصیات **Display** پشتیبانی می‌کند.

opacity: افزونه‌ی **CSS** ای **moz-opacity** موزیلا، در ازای خصوصیت استاندارد **opacity** حذف شد.

text-shadow: این خصوصیت که امکان سایه دادن به محتویات متنی وب را فراهم می‌سازد، توسط فایر فاکس ۳٫۵، پشتیبانی


```
faces.nsILoadContext);
}
catch (ex)
{
    try
    {
        loadContext =
            aRequest.loadGroup.notificationCallbacks.
                getInterface(Components.interfaces.nsILoadContext);
    }
    catch (ex)
    {
        loadContext = null;
    }
}
```

به علاوه رفتار نوار ابزار های سفارشی، در فایر فاکس ۳٫۵ تغییر کرده؛ به عنوان مثال `</xul:toolbar>` باعث حذف بخش های نوار از مجموعه اش می شود و `</xul:toolbarpalette>` به جای کپی گرفتن از آنها و اعمال آن به نوار ابزار، مستقیم به نوار اضافه می کند. این بدان معنی است که palette دیگر تنها شامل عناصری است که در نوار ابزار نیستند بر خلاف گذشته که کلیه ی عناصر را شامل می شد. البته این کار می تواند باعث بروز مشکلاتی در addons ها بسته به این که بتوان کلیه ی نوار ابزارهای سفارشی را توسط `</xul:toolbarpalette>` برگرداند یا نه، می شود. به علاوه زمانی که بخشی را به صورت پویا به palette برای دسترسی از طریق نوار ابزار سفارشی اضافه می کنیم نیز، این مشکل احتمال بروز دارد. حال که به یک باگ اشاره شد، جا دارد که به حفره ی امنیتی هم که به خاطر ریموت کروم، به وجود آمده بود اشاره کنیم. بدین وسیله هر افزونه ای که منبعی در فایل chrome.manifest داشت به صورتی که به یک سایت اشاره می کرد، تحت تأثیر قرار می گرفت و دچار تغییراتی می شد. این مشکل، در نسخه ی جدید فایر فاکس بر طرف شده و دیگر تأثیری بر روی افزونه ها نخواهد داشت.

• و در نهایت، گزینشگرهای پرس و جویی جاوا اسکریپت^۴، مبدل های SVG^۵ و برنامه های آفلاین^۶ و رفع یک مشکل امنیتی تحت عنوان مخرب وضعیت JIT در نسخه ی ۳٫۵٫۱ که می توانست به حفره ای برای نفوذ مهاجمان برای نصب برنامه های Malware توسط آن ها شود. این ها همگی از امکانات جدید فایر فاکس ۳٫۵ است که باعث هر چه قدرتمند تر و سریع تر شدن این مرورگر شده است.

```
places.append("places.sqlite");

var db = Components.
    classes["@mozilla.org/storage/
        service;1"].
        getService(Components.interfaces.
            mozIStorageService).
            openDatabase(places);
```

بدین وسیله مسیری به دیتابیس places.sqlite ساخته می شود و سپس فایل برای دسترسی انباره، باز می شد. در فایر فاکس ۳٫۵ سرویسی اضافه شده که راه ساده تری برای دسترسی به دیتابیس Places ارائه می کند:

```
var db = Components.
    classes["@mozilla.org/browser/
        nav-history-service;1"].
        getService(Components.interfaces.
            nsPIPlacesDatabase).
            DBConnection;
```

همچنین در نسخه های قبلی برای بدست آوردن متن لود یک درخواست، از API های متفاوت DocShell استفاده می شد، لکن راه ساده تر و آسان تری که در فایر فاکس ۳٫۵ پیشنهاد می شود استفاده از nsILoadContext است:

راه کار موجود در ++C:

```
nsCOMPtr<nsILoadContext> loadContext;
nsCOMPtr<nsIChannel> channel = do_QueryInterface(aRequest);
NS_QueryNotificationCallbacks(channel,
    loadContext);
```

و در JavaScript:

```
var loadContext;
try
{
    loadContext =
        aRequest.
            queryInterface(Components.interfaces.
                nsIChannel).
                notificationCallbacks.
                    getInterface(Components.interfaces.
```

۱- Shiretoko

۲- Web Workers

۳- W3C Geolocation API

۴- JavaScript Query Selectors

۵- SVG Transforms

۶- Offline Applications

۷- Downloadable Fonts Support

۸- CSS Media Quires

۹- Media-Dependent Style Sheets



انستیتو ایزایران

آموزشگاه علمی - تخصصی رایانه نبی اکرم (ص) - نمایندگی شرق

برگزار کننده دوره های کاربردی و حرفه ای مطابق با استانداردهای جهانی
و هماهنگ با گواهینامه های بین المللی و تخصصی

این آموزشگاه با بهره گیری از اساتید مجرب و کار آزموده و نیز امکانات آموزشی کامل، لیست دوره های خود
را به شرح ذیل اعلام می نماید:

سیستم های عامل شامل Windows XP & Vista and Linux

دوره های مهارتی ICDL

دوره های فنی و مهندسی مانند AutoCad, Solidworks و ...

تجزیه و تحلیل آماری و کنترل پروژه مانند Spss و دیگر نرم افزارها

دوره های مقدماتی و پیشرفته نرم افزارهای مجموعه Office مانند Word, Excel و ...

زبانهای برنامه نویسی مانند NET. در کلیه گرایشها، Delphi، برنامه نویسی تحت برنامه های Office و ...

دوره های طراحی و برنامه نویسی وب مانند ASP, Net, PHP, CIW و ...

دوره های تخصصی شبکه مانند MCSE, MCSA, Network+ و دوره های CISCO

بانکهای اطلاعاتی مانند SQL Server, Oracle و ...

دوره های گرافیک مانند In Design, Corel Draw, Photoshop و ...

دوره های سخت افزاری سیستم مانند تعمیر و نگهداری سخت افزار و ...

آدرس: نارمک - ۴۶ متری غربی (شهید ثانی) - نبش سمندگان
تلفن: ۷۷۲۵۵۳۵۰-۱

نویسنده: محمد جاهد منش

نصب و راه اندازی یک Mail Server کامل با qmail

قسمت اول شامل: تعاریف، پیش نیازها، نصب Qmail، نصب DNS Server

xvay.com@gmail.com



مرحله ۳: نصب پکیج‌های مورد نیاز

```
yum -y install autoconf automake automake17 bzip2 bzip2-devel
bzip2-libs compat-gcc-34 compat-gcc-34-c++ compat-glibc compat-glibc-headers
compat-libf2c compat-libgcc compat-libstdc++-296 compat-libstdc++-33 curl curl-devel
expect expect-devel gcc gcc-c++ gdbm gdbm-devel gmp gmp-devel httpd httpd-devel
httpd-manual krb5-auth-dialog krb5-devel krb5-libs krb5-workstation libgcc libidn
libidn-devel libtool libtool-ltdl libtool-ltdl-devel mysql mysql-bench mysql-devel
mysql-server mrtg ncurses-devel ntp openssh openssh-clients openssh-askpass
openssh-server openssl openssl-devel pcre pcre-devel perl-Digest-HMAC perl-Digest-SHA1
perl-HTML-Parser perl-libwww-perl perl-Net-DNS php php-ldap php-mysql php-pear
redhat-rpm-config rpm rpm-build rpm-devel rpm-libs rpm-python sed setup
setuptools stunnel system-config-date wget which xinetd zlib zlib-devel
```

تعاریف:

qmail نرم‌افزاری open source برای ایجاد یک Mail Server بر روی بستر لینوکس که قابلیت پشتیبانی از SMTP, POP3, IMAP و ... را دارد و همچنین به همراه پکیج‌های مدیریتی بسیاری ارائه شده است که می‌توان به web mail, mrtg و webadmin اشاره کرد.

تنظیمات DNS جزء جدا نشدنی یک میل سرور کامل و سالم که می‌بایست بدون کوچک‌ترین عیب و ایرادی تنظیم شود، به این علت که میل سرورهای دیگر از جمله yahoo, google, live و ... به طور جدی نسبت به تنظیمات DNS حساسیت نشان می‌دهند و مانع ورود ایمیل‌هایی بدون تنظیمات DNS صحیح خواهند شد.

Domainkey: یک کلید برای احراز هویت نام Domain که ابداعی از yahoo هست و به منظور جلوگیری هرچه بیشتر از Spam ایجاد شده.

می‌توانید مطالب بیشتر را در آدرس <http://antispam.yahoo.com/domainkeys> می‌باشد پیگیری کنید.

DKIM: ابداعی دیگر که کارایی همانند Domainkey دارد. مطالب بیشتر در این خصوص را در DKIM.org پیگیری نمایید.

پیش نیازها:

مرحله ۱: یکی از توزیع‌های لینوکس. این مقاله بر اساس توزیع Centos ۵ تهیه شده است.

مرحله ۲: حذف پکیج‌های نا سازگار

```
yum -y remove sendmail
yum -y remove openssl
yum -y remove cyrus-imapd
```

مرحله ۴: نصب پکیج های Perl برای Spamassassin

```

کد:
perl -e 'use CPAN; install MIME::
Base64;'
perl -e 'use CPAN; install DB_File;'
perl -e 'use CPAN; install Net::
DNS;'
perl -e 'use CPAN; install Net::
SMTP;'
perl -e 'use CPAN; install Mail::
SPF::Query;'
perl -e 'use CPAN; install Time::
HiRes;'
perl -e 'use CPAN; install Mail::
DomainKeys;'
perl -e 'use CPAN; install IO::
Zlib;'
perl -e 'use CPAN; install Archive::
Tar;'

```

مرحله ۵: Stop سرویس های غیر ضروری و استارت سرویس های ضروری.

```

chkconfig httpd on
service httpd start

chkconfig mysqld on
service mysqld start

chkconfig ntpd on
service ntpd start

```

مرحله ۶: تنظیمات mysql

```

mysqladmin -uroot -prootpassword
mysqladmin -uroot -prootpassword
reload
mysqladmin -uroot -prootpassword
refresh

```

ایجاد دیتابیس برای vpopmail

```

mysqladmin create vpopmail -uroot -
prootpassword
mysqladmin -uroot -prootpassword
reload
mysqladmin -uroot -prootpassword
refresh

echo "GRANT ALL PRIVILEGES ON
vpopmail.* TO vpopmail@localhost
IDENTIFIED BY 'vpopmailpassword'" |
mysql -uroot -prootpassword
mysqladmin -uroot -prootpassword
reload
mysqladmin -uroot -prootpassword
refresh

```

مرحله ۷: runlevel

```

cp -u /etc/inittab /etc/inittab.bak
cat /etc/inittab | sed -e 's/^id:۵:
initdefault:/id:۳:initdefault:/' >
/etc/inittab.new
mv -f /etc/inittab.new /etc/inittab

```

مرحله ۸: ایجاد symbol link برای com_err.h

```

ln -s /usr/include/et/com_err.h /usr/
include/com_err.h

```

مرحله ۹: به روز رسانی مخزن yum

```

yum -y update

```

مرحله ۱۰: ریست سیستم

```

reboot

```

نصب Qmail:

مرحله ۱: دریافت پکیج های مربوط


```
mkdir -p /usr/src/qtms-install
cd /usr/src/qtms-install

wget http://www.qmailtoaster.com/download/zlib-1.2.3-1.0.3.src.rpm
wget http://www.qmailtoaster.com/download/daemontools-toaster-0.76-1.3.4.src.rpm
wget http://www.qmailtoaster.com/download/ucspi-tcp-toaster-0.88-1.3.6.src.rpm
wget http://www.qmailtoaster.com/download/vpopmail-toaster-5.4.17-1.3.5.src.rpm
wget http://www.qmailtoaster.com/download/libdomainkeys-toaster-0.68-1.3.4.src.rpm
wget http://www.qmailtoaster.com/download/libsrss2-toaster-1.0.18-1.3.4.src.rpm
wget http://www.qmailtoaster.com/download/qmail-toaster-1.03-1.3.16.src.rpm
wget http://www.qmailtoaster.com/download/courier-authlib-toaster-0.59.2-1.3.7.src.rpm
wget http://www.qmailtoaster.com/download/courier-imap-toaster-4.1.2-1.3.8.src.rpm
wget http://www.qmailtoaster.com/download/autorespond-toaster-2.0.4-1.3.4.src.rpm
wget http://www.qmailtoaster.com/download/control-panel-toaster-0.5-1.3.5.src.rpm
wget http://www.qmailtoaster.com/download/ezmlm-toaster-0.53.324-1.3.4.src.rpm
wget http://www.qmailtoaster.com/download/qmailadmin-toaster-1.2.11-1.3.5.src.rpm
wget http://www.qmailtoaster.com/download/qmailmrtg-toaster-4.2-1.3.4.src.rpm
wget http://www.qmailtoaster.com/download/maildrop-toaster-2.0.3-1.3.6.src.rpm
wget http://www.qmailtoaster.com/download/isoqlog-toaster-2.1-1.3.5.src.rpm
wget http://www.qmailtoaster.com/download/squirrelmail-toaster-1.4.17-1.3.12.src.rpm
wget http://www.qmailtoaster.com/download/spamassassin-toaster-3.2.5-1.3.15.src.rpm
wget http://www.qmailtoaster.com/download/clamav-toaster-0.95.1-1.3.27.src.rpm
wget http://www.qmailtoaster.com/download/ripmime-toaster-1.4.0.6-1.3.4.src.rpm
wget http://www.qmailtoaster.com/download/simscan-toaster-1.3.1-1.3.7.src.rpm
wget http://www.qmailtoaster.com/download/vqadmin-toaster-2.3.4-1.3.4.src.rpm
wget http://www.qmailtoaster.com/download/djbdns-1.05-1.0.6.src.rpm
```

مرحله ۲: نصب پکیج ها

```
wget http://www.qmailtoaster.com/centos/cnt50/cnt50-install-script.sh
sh cnt50-install-script.sh
```

در این مرحله می توانید تمام مراحل را با استفاده از **Enter** نصب نمایید.

توجه : نصب همه پکیج ها الزامی است در صورتی که از **DNS Server bind** استفاده می کنید در این صورت از نصب پکیج **1.0.6.src.rpm-djbdns-1.05** جلوگیری کنید تا دچار تداخل نشوید.

توجه : در هنگام نصب پکیج ها در صورت وجود هرگونه تداخل از نصب پکیج جلوگیری می شود.

مرحله ۳: تنظیم سرویس های **startup**

```
chkconfig acpid on
chkconfig anacron on
chkconfig atd on
chkconfig autofs on
chkconfig cpuspeed on
chkconfig crond on
chkconfig freshclam on
chkconfig haldaemon on
chkconfig httpd on
chkconfig iptables on
chkconfig kudzu on
chkconfig messagebus on
chkconfig mysqld on
chkconfig network on
chkconfig ntpd on
chkconfig qmail on
chkconfig smartd on
chkconfig sshd on
chkconfig syslog on
chkconfig xinet on
chkconfig irqbalance on
```

نصب **DNS Server**

در این مقاله از **bind** استفاده شده است.

مرحله ۱: نصب پکیج های مورد نیاز

```
yum -y install bind bind-chroot
bind-devel bind-libbind-devel bind-libs
bind-utils ypbind caching-name-server
```

مرحله ۲: تنظیم **DNS Server** داخلی

```
echo "search your-domain.com" > /etc/resolv.conf

echo "nameserver 127.0.0.1" >> /etc/resolv.conf
```

مرحله ۳: **Start Up Dns Server**

```
chkconfig named on
```

مرحله ۴: ریست سیستم

```
reboot
```

آزمایش صحت نصب برنامه :

```
qmailctl stat
```

و در جواب داریم :

```
authlib: up (pid 2425) 65 seconds
clamd: up (pid 2425) 65 seconds
imap4: up (pid 2421) 65 seconds
imap4-ssl: up (pid 2423) 65 seconds
pop3: up (pid 2414) 65 seconds
pop3-ssl: up (pid 2409) 65 seconds
send: up (pid 2416) 65 seconds
smtp: up (pid 2418) 65 seconds
spamd: up (pid 2407) 65 seconds
authlib/log: up (pid 2417) 65 seconds
clamd/log: up (pid 2417) 65 seconds
imap4/log: up (pid 2422) 65 seconds
imap4-ssl/log: up (pid 2424) 65 seconds
pop3/log: up (pid 2415) 65 seconds
pop3-ssl/log: up (pid 2413) 65 seconds
send/log: up (pid 2420) 65 seconds
smtp/log: up (pid 2419) 65 seconds
spamd/log: up (pid 2408) 65 seconds
```

Web Mail :

برای استفاده از webmail می توانید از آدرس `http://mydomain.com/webmail` استفاده نمایید.

توجه کنید! اگر طبق این مقاله پیش آمده باشید در وضعیت فعلی تنها از طریق وب میل قادر به ارسال ایمیل به دیگر میل سرور ها هستید، تنظیمات مربوط به Relay IP در قسمت های بعدی توضیح داده خواهد شد

بخش سوم : توضیحاتی در مورد تنظیمات DNS Server

در سال های اخیر با گسترش هرز نامه ها، میل سرورهای محبوب هر کدام به نوبه خود تدابیری برای جلوگیری از این دسته از ایمیل ها اندیشیده اند.

به عنوان مثال تمام میل سرورها ایمیل های ورودی از یک IP را یا Block و یا به قسمت Spam هدایت می کنند.

پس دیگر نصب یک میل سرور local و ارسال ایمیل عملاً بی معنی و بلا استفاده در شبکه اینترنت است.

بعد از مدتی متقلبان دیگری با استفاده از نام دامین های دیگر اقدام به ارسال هرزنامه می کردند که برای جلوگیری از این مسئله نیز تدابیری اندیشه شد که می توان به تنظیمات DNS Server اشاره کرد.

البته نکته قابل اشاره این مسئله هست که در ابتدای راه اندازی میل سرور ایمیل های شما به درستی به مقصد خواهد رسید ولی بعد از مدتی سخت گیری ها آغاز خواهد شد و در صورتی که اطلاعی از مسئله تنظیمات DNS نداشته باشید کاملاً سر در گم خواهید شد.

این گونه سخت گیری ها برای میل سرورها روش های مختلفی دارد که ما در این مقاله به ۳ میل سرور بزرگ و محبوب اشاره خواهیم داشت که شما با پوشش صحیح این ۳ میل سرور می توانید مطمئن باشید که میل سرور شما با تمام میل سرورهای دیگر سازگار خواهد بود.

live.com : نیاز به تنظیمات صحیح و کامل DNS دارد.
gmail.com : نیاز به ست کردن SPF Record در DNS Server دارد.

yahoo.com : علاوه بر تمام موارد بالا نیاز به Domainkey و DKIM دارد.

(در صورتی که با ست کردن صحیح این گزینه ها باز هم نامه های ارسالی از میل سرور شما به قسمت Spam رفت نیاز به مکاتبه با مسئولین یاهو دارید که مفصلاً شرح داده خواهد شد.)

بخش چهارم : live.com تنظیمات کامل و صحیح DNS Server

هر دامنه ای حداقل دارای 1 DNS می باشد به عنوان مثال

بخش دوم : افزودن Domain و User و تنظیمات webmail مدیریت web

با استفاده از دستورات زیر می توانیم دامین خود که در این مقاله `mydomain.com` نام گذاری شده را تعریف و همچنین کاربران هر دامین را تعریف نماییم.

افزودن دامین جدید :

```
/home/vpopmail/bin/vaddomain mydo-  
main.com <postmaster-password>
```

توجه کنید: برای هر دامین که افزوده می شود یک آدرس با نام `postmaster` ایجاد می گردد که `password` آن به جای `<postmaster-password>` قرار می گیرد.

افزودن User جدید :

```
/home/vpopmail/bin/vadduser  
info@mydomain.com <email-password>
```

تنظیمات Web Interface :

ابتدا `php.ini` را واقع در `etc` ویرایش می کنیم و `register_globals` را برابر با `On` قرار می دهیم سپس سرویس `httpd` را ریستارت می کنیم.

```
vi /etc/php.ini  
  
#Now Set  
register_globals = On  
#Now  
service httpd restart
```

حال می توانید `interface` خود را در آدرس `http://mydomain.com/admin-toaster` مشاهده کنید.

نام کاربری `admin`

پسورد پیش فرض `toaster`

در این اینترفیس می توانید دامین اضافه کنید، کاربر برای هر دامین اضافه کنید، گزارشی از ایمیل های ورودی و خروجی از سرور داشته باشید و همچنین `mrtg` سرور خود را مشاهده کنید.

توجه : در صورتی که `Isoqlog` را مشاهده نکردید باید اسکریپت زیر را اجرا نمایید.

```
sh /usr/share/toaster/isoqlog/bin/  
cron.sh
```

```
mail      IN      A      <mailserverip>
@         IN      MX      10      mail.my-
domain.com.
```

در مرحله بعدی نیاز به PTR Record n دارید که اشاره‌گری از ip شما به نام رکورد A برای میل سرور هست. پس در zone مربوط به دامنه داریم:

```
x.x.x.n   IN      PTR      mail.mydo-
main.com.
```

و در zone برای in-addr.arpa داریم:

```
$ORIGIN x.x.x.IN-ADDR.ARPA.
@         IN      SOA      ns1.mydo-
main.com. postmaster.mydomain.com. (

    2003080800 ; sn = serial number
                                172800
; ref = refresh = 2d
                                900
; ret = update retry = 15m

    1209600    ; ex = expiry = 2w
                                3600
; min = minimum = 1h
)

        IN      NS      ns1.
mydomain.com.

n        IN      PTR      mail.
mydomain.com.
```

توجه! در مثال بالا منظور از x.x.x آدرس network و منظور از n شماره هاست می باشد.

حال DNS Server خود را update کنید و دامین خود را با <http://www.dnscheck.se> یا دیگر ابزار های کنترل DNS چک کنید و تا جایی پیش بروید که هیچگونه ایرادی در DNS

برای دامنه ما که با نام mydomain.com می شناسیم ns1.mydomain.com وجود دارد و همچنین برای هر دامین یک ایمیل postmaster یا hostmaster تعریف شده است که باید در SOA Record تعریف شده باشد. توجه کنید که در هنگام ایجاد دامین در Qmail اکانت Postmaster ایجاد خواهد شد. پس داریم:

```
@         IN      SOA      ns1.mydomain.com.
postmaster.mydomain.com. (

    2003080800 ; sn = serial number
                                172800
; ref = refresh = 2d
                                900
; ret = update retry = 15m

    1209600    ; ex = expiry = 2w
                                3600
; min = minimum = 1h
)
```

در مرحله بعد نیاز به NS Record داریم تا نام DNS خود را برای DNS Server تعریف کرده باشیم، پس داریم:

```
@         IN      NS      ns1.mydomain.
com.
```

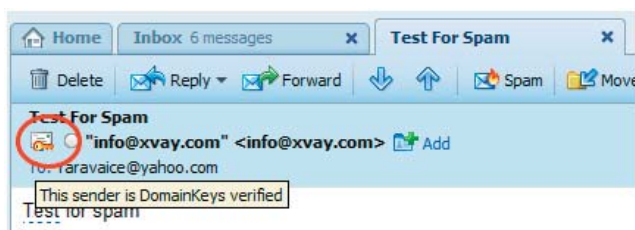
در مرحله بعد نیاز داریم تا هرکدام از NS هایی که تعریف کرده ایم را توسط A Record هاست کنیم همچنین postmaster یا hostmaster که در رکورد SOA تعریف شده اند نیز باید هاست شده باشند. پس داریم:

```
ns1      IN      A      <ipaddress>
hostmaster IN      A
<ipaddress>
```

در مرحله بعد نیاز داریم که میل سرور خود را نیز هاست کنیم و رکورد MX که در اصل راهنمایی برای دیگر میل سرورها برای پیدا کردن میل سرور در ip شما هست اضافه کنیم. پس داریم:

اطمینان پیدا می کند.

در ایمیل هایی که ارسال کننده آن دارای Domainkey معتبر باشند به صورت پیش فرض توسط یاهو به کاربر اعلام می گردد.



روش نصب و ایجاد Domainkey برای دامین فرضی mydomain.com

مرحله اول ایجاد Folder برای Key

```
cd /var/qmail/control/domainkeys
mkdir mydomain.com
```

مرحله دوم ساخت جفت کلید Public و Private :

```
cd mydomain.com
dknewkey private > public.txt
```

بعد از اجرای دستور بالا در داخل پوشه mydomain.com دو فایل با نام های private و public.txt ایجاد خواهد شد. نا گفته نماند که انتخاب نام برای این ۲ فایل اختیاری می باشد، محتوای فایل public.txt حاوی تنظیمات DNS است که به صورت کلید عمومی در DNS سرور قرار می گیرد. به عنوان مثال

```
private._domainkey IN TXT
"k=rsa; p=MEwwDQYJKoZIhvcNAQEBBQADAwOAIXAN6Go..."
```

توجه قسمت پر رنگ در مثال بالا همان کلید عمومی می باشد که با پارامتر p مشخص شده و پارامتر k نشان دهنده نوع کلید می باشد. و فایل private شامل کلید عمومی که توسط qmail به header هر ایمیل ارسالی الحاق خواهد شد. توجه کنید که انتخاب نام برای این فایل لازم به توجه بیشتری است زیرا همین نام باید در DNS نیز تعریف گردد.

حال می بایست پرمیشن های لازم برای پوشه و دو فایل مذکور تعیین گردد، بدین منظور از دستورات زیر استفاده خواهیم کرد :

سرور شما شناسایی نشود. (در این سایت ایراد ها به رنگ قرمز نشان داده خواهند شد)

همچنین برای تست کردن ptr record از ابزار موجود در : <http://www.zoneedit.com/lookup.html>

و برای تست mx record از ابزار موجود در :

<http://www.mxtoolbox.com> استفاده کنید.

خسته نباشید! اکنون ایمیل های شما در live.com به خوبی دریافت خواهند شد.

بخش پنجم : gmail.com و افزودن SPF Record

در gmail شما نیاز دارید که علاوه بر رعایت سلامت DNS Server خود که در بخش قبلی به آن اشاره شد IP میل سرور خود را در DNS Server مربوط به دامین ارسال کننده را احراز هویت کنید که این کار توسط یک TXT Record با محتوی spf انجام می شود.

بدین منظور می بایست این رکورد به zone مربوط اضافه شود.

```
mydomain.com. IN TXT
"v=spf1 ptr:mail.mydomain.com mx:
mail.mydomain.com ip4:<SMTP-ip
Address> ~all"
```

به منظور افزودن و یا تغییرات دلخواه می توانید از ابزار wizard در آدرس <http://openspf.org> استفاده کنید.

همچنین به منظور تست spf می توانید از ابزار :

<http://www.kitterman.com/spf/validate.html> یا

ابزارهایی همانند nslookup یا dig استفاده کنید.

راه دریافت ایمیل های شما برای کاربران gmail.com هموار است.

بخش ششم : yahoo.com

• قسمت اول Domainkey

در yahoo علاوه بر تنظیمات صحیح DNS Server و همچنین SPF Record نیاز به Domainkey نیز وجود دارد.

Domainkey چیست؟

یکی از ابداعات یاهو برای جلوگیری از Spam و احراز هویت دامین ارسال کننده توسط یک جفت کلید ssl که کلید Private در سرایند ایمیل ارسال می شود و کلید Public در DNS Server قرار می گیرد و بدین وسیله یاهو از هویت ارسال کننده نامه



SSL Certificate

جهت کسب اطلاعات بیشتر و مشاهده جزئیات محصولات به وب سایت زیر مراجعه نمایید:

<http://parmidaco.net>

شرکت پیشرو ارتباط پارمیدا



```
chmod 440 private
cd ..
chown -R root:vchkw mydomain.com
```

تنظیمات DNS برای Domainkey :

گام اول در تنظیمات DNS Server ایجاد یک TXT Record با نام پیش فرض _domainkey می باشد که به شرح زیر است :

```
_domainkey.your-domain.
com. IN TXT "t=y; o=-;
r=postmaster@mydomain.com"
```

توجه کنید که t=y به این معنی است که این Record در حال تست می باشد و با استفاده از ابزارهایی که تعریف خواهد شد می توانید DNS Server خود را برای Domainkey تست کنید و در پایان راه اندازی می توانید t=y را حذف کنید. گام دوم ایجاد یک TXT Record برای کلید عمومی که در حقیقت همان محتوای public.txt می باشد :
کد:

```
private._domainkey IN TXT
"k=rsa; p=MEwwDQYJKoZIhvcNAQEBBQADow
AwOAIxAN\Go..."
```

توجه برای private._domainkey :

همانطور که مشاهده کردید این رکورد با نام private در زیر مجموعه _domainkey نام گذاری شده است که در حقیقت private همان نامی است که هنگام ایجاد کلید توسط شما انتخاب شده!

هنگامی که qmail کلید private را به سرآیند ایمیل الحاق می کند نام فایل کلید خصوصی نیز برای yahoo ارسال می شود و yahoo با استفاده از آن نام در زیر مجموعه _domainkey از دامین شما به دنبال کلید خصوصی خواهد گشت. پس دقت کنید که این نام و نام فایل کلید خصوصی باید یکسان باشد.

بعد از به روز رسانی DNS Server می توانید از طریق دو آدرس زیر Domainkey خود را تست کنید :

<http://domainkeys.sourceforge.net/policycheck.html>

<http://domainkeys.sourceforge.net/selectorcheck.html>

نگاهی اجمالی به هوک‌های ویندوز

چکیده: استفاده از هوک، یکی از مباحث تکنیکی مطرح در زمینه برنامه‌نویسی سیستمی بوده که کاربرد بسیاری در ساخت برنامه‌های کنترل‌کننده سیستم عامل دارد. هوک به معنی ایجاد شود بر روند اجرای وقایعی مانند فراخوانی توابع، ارسال پیام‌ها و پاسخ به رخدادها در سیستم عامل می‌باشد. از این تکنیک در ساخت نرم‌افزارهایی مانند آنتی‌ویروس‌ها، فایروال‌ها، دیباگرها، نرم‌افزارهای جاسوسی و ویروس‌ها استفاده می‌شود. همچنین در این مقاله روش‌هایی جهت شنود بر توابع API معرفی و تشریح شده و در پایان، ساختار فایل‌های اجرایی ویندوز و نحوه اجرای آن‌ها در سیستم عامل مورد بررسی قرار می‌گیرد.

<http://www.mshams.ir>

کلمات کلیدی

هوک؛ شنود توابع؛ API سیستم پیام‌رسان؛ تزریق کد؛ ساختار فایل PE

بر تمامی انواع پیام‌های سیستم عامل به سادگی امکان پذیر نیست. ایده اصلی هوک، در واقع نوشتن تابعی است که به محض فعال شدن رویدادی خاص (Event) در سیستم عامل، اجرا می‌شود. حال این تابع می‌تواند فقط شاهد داده‌های ارسالی بوده و یا آن‌ها را تغییر هم بدهد.

۱-۱- فواید استفاده از هوک

- تغییر مکانیسم یا روند اجرای برنامه‌ها: با استفاده از هوک می‌توان اعمال دلخواهی، قبل و یا بعد از فعال شدن رویدادهای برنامه انجام داد. بدین وسیله می‌توان امکاناتی به برنامه‌های آماده و کامپایل شده افزوده و یا روند پیش‌فرض آن‌ها را تغییر داد. مثلاً جلوگیری از عملی که یک قفل نرم‌افزاری برای بررسی وجود CD در درایو انجام می‌دهد.

- دیباگ و مهندسی معکوس: معمول‌ترین روش دیباگ کردن برنامه‌های کامپایل شده استفاده از هوک‌های سیستم عامل است. با این کار می‌توان رابطه مابین بخش‌های مختلف یک نرم‌افزار و نحوه تعامل آن‌ها با سیستم عامل را بررسی کرده و به ایرادهای احتمالی آن پی برد. به این عمل اصطلاحاً مهندسی معکوس گفته می‌شود.

- کشف قابلیت‌های مخفی سیستم عامل: تعداد زیادی از توابع API موجود در ویندوز، مخفی بوده و هیچ گونه مستنداتی از طرف شرکت مایکروسافت جهت معرفی آن‌ها ارائه نشده است که اصطلاحاً به آن‌ها Undocumented API گفته می‌شود.

در حال حاضر کتاب‌های بسیاری به این مقوله پرداخته که حاصل مدت‌ها تحقیق و بررسی و استفاده از روش‌های مهندسی معکوس بر ساختار سیستم عامل ویندوز می‌باشند. در این کتب، تعداد زیادی از توابع مستند نشده ویندوز معرفی شده و نحوه کار آن‌ها تشریح شده است.

یکی از نکاتی که قبل از استفاده از هوک می‌بایست به آن توجه

مقدمه: یکی از مباحث مهمی که در زمینه برنامه‌نویسی سیستمی مطرح می‌شود، کنترل دقیق نحوه انجام کارها توسط سیستم عامل است. بدین منظور، برنامه‌نویسان اقدام به نصب و راه‌اندازی شنودگرهای نرم‌افزاری جهت کنترل نحوه پاسخ‌دهی سیستم عامل به وقایع سیستم می‌کنند. بدیهی است که در حین کنترل این روند، می‌توان تغییراتی دلخواه، بسته به نیاز و هدف برنامه‌نویس، در نحوه انجام آن اعمال کرد.

۱- هوک چیست؟

همانطور که می‌دانیم سازوکار سیستم عامل ویندوز، مبتنی بر پیام است. یعنی تمام اعمالی که در ویندوز انجام می‌شود از طریق ارسال پیام‌هایی از بخش‌های مختلف این سیستم عامل به سایر بخش‌ها صورت می‌گیرد. به عنوان مثال وقتی که پنجره‌ای را در ویندوز تغییر مکان می‌دهیم، پیامی توسط بخش کنترل‌کننده اینترفیس برنامه، حاوی اطلاعات موقعیت پنجره و از طریق تابع API به نام SendMessage() به بخشی از هسته سیستم عامل ارسال گشته و سیستم عامل بر اساس پارامترهای دریافتی، کار مورد نظر را انجام می‌دهد. با توجه به مکانیسم ذکر شده، به عمل نصب شنودگرها در سیستم پیام‌رسان سیستم عامل، هوک گفته می‌شود [۱].

احتمالاً در ابتدا هوک‌ها جهت استفاده در دیباگرها و اعمال کنترلی سیستم معرفی و به وجود آمده بودند اما امروزه، برنامه‌نویسان کاربردهای فراوانی برای آن‌ها پیدا کرده‌اند. مثلاً برنامه‌ای برای ثبت کلیدهای فشرده شده در سیستم.

هوک‌ها بر روی انواع خاصی از پیام‌های ارسالی نصب شده و شنود

می‌دهد افزایش می‌یابد و در این حین توابع مربوط به تمام هوک‌ها می‌بایست تک‌تک پردازش شده تا در نهایت پیام مربوط به مقصد نهایی برسد. در نتیجه تنها زمانی از هوک‌ها استفاده می‌شود که راه دیگری وجود نداشته باشد. همچنین اگر کوچکترین خطایی در حین کار برنامه سرویس دهنده یا درایور آن رخ دهد، ممکن است روند اجرای سیستم عامل مختل گردد.

پس از نصب هوک سراسری، ویندوز فایل DLL مربوطه را در فضای آدرس (Address space) مربوط به تمام process‌ها نوشته و آن را به عنوان قسمتی از روند ارسال پیام، مسیردهی می‌کند. (شکل ۱)

پس از پایان عمل شنود و یا زمانی که نیاز به خروج از برنامه سرویس دهنده باشد، هوک نصب شده باید از زنجیره هوک‌ها خارج شود. برای این کار از تابع UnHookWindowsHookEx() استفاده می‌شود. این تابع اشاره‌گر دریافت شده از SetWindowsHookEx را به عنوان پارامتر ورودی دریافت کرده و اقدام به حذف هوک می‌کند.

هنگامی که این تابع فراخوانی می‌شود، سیستم عامل صبر می‌کند تا تمام process‌هایی که در حال استفاده از تابع هوک هستند پردازش خود را تمام کرده و سپس اقدام به حذف هوک از فضای آدرس process‌ها می‌نماید. ساختار توابع ذکر شده در جدول ۱ تا ۶ تشریح شده است.

۱-۲ پیام‌های مورد استفاده در هوک WH_CALLWNDPROC: شنود بر پیام‌هایی که با استفاده از تابع SendMessage به پنجره‌ها ارسال می‌شوند. نوع تابع آن CallWndProc بوده و درست قبل از ارسال پیام ذکر شده اجرا می‌گردد.

WH_CALLWNDPROCRET: پس از پردازش پیام‌های مذکور در برنامه پنجره مقصد، توسط این هوک می‌توان آنها را مشاهده کرد. نوع تابع آن CallWndProcRet است. WH_CBT: نوع تابع خروجی CBTProc بوده و قبل از اعمالی مثل فعال شدن پنجره‌ها، ساخت، تخریب، تغییر مکان یا اندازه پنجره‌ها و یا تمرکز روی کنترل‌ها (Focus) و به عبارتی قبل از پردازش هر عملی که توسط ماوس یا کیبورد بر روی پنجره‌ها انجام می‌شود، اجرا می‌گردد.

WH_DEBUG: نوع تابع آن DebugProc است و دقیقاً قبل از فراخوانی توابع هوک نصب شده در سیستم اجرا شده و اطلاعاتی در مورد هوک فعال شده دریافت می‌کند. به عبارتی به منظور دیباگ کردن روند اجرای دیگر هوک‌ها از آن استفاده می‌شود.

WH_GETMESSAGE: نوع تابع آن GetMessage بوده و دقیقاً زمانی اجرا می‌شود که برنامه‌ای با استفاده از توابع

شود این است که هدف، شنود بر یک برنامه خاص بوده یا به منظور بررسی عملی، در کل سیستم عامل است.

مثلاً اگر بخواهیم پارامترهای ارسالی در تابع SendMessage به پنجره‌های ویندوز را به دست آوریم، می‌بایست یک هوک سراسری (Global hook) نصب شود، اما اگر هدف فقط بررسی یک برنامه تنها باشد نیازی به این کار نبوده و از یک هوک محلی (Local hook) استفاده می‌شود. به عبارتی یک هوک سراسری، رویدادهای تمام process‌های موجود را شنود می‌کند.

۲- ساختار کلی هوک

از دو بخش سرویس دهنده (Hook server) و درایور تشکیل می‌شود. درایور کار اصلی را انجام داده و پیام‌های مورد نظر را از سیستم دریافت می‌کند. سرور نیز در تعامل با درایور بوده و بنا به خواست برنامه‌نویس، برای نحوه برخورد با پیام‌ها تصمیم می‌گیرد. در یک سیستم هوک سراسری، درایور در یک فایل DLL و کاملاً مجزا از قسمت سرور قرار می‌گیرد. به عبارتی برنامه سرور اقدام به راه‌اندازی درایور از درون فایل DLL می‌نماید.

به طور دقیق‌تر سرور می‌بایست با استفاده از تابع LoadLibrary و به طور پویا، اقدام به بارگذاری فایل DLL نموده و Handle آن را به دست آورد. سپس با استفاده از تابع GetProcAddress اشاره‌گر مربوط به تابع نصب هوک را به دست آورده و از آن استفاده کند. با این کار درایور با استفاده از تابع SetWindowsHookEx تابع هوک را به عنوان اولین حلقه از زنجیره هوک‌های سیستم نصب می‌کند.

تابع SetWindowsHookEx اشاره‌گر مربوط به آدرس هوک نصب شده را به عنوان مقدار بازگشتی برمی‌گرداند. این مقدار اهمیت زیادی دارد زیرا می‌بایست با استفاده از تابع CallNextHookEx این آدرس را به ادامه زنجیره هوک‌ها متصل کنیم. همچنین پس از پایان کار برای حذف هوک از این آدرس استفاده می‌شود [۲].

در مورد هر نوع از پیام‌های ارسالی در سیستم عامل، ممکن است تعداد زیادی هوک به شکل یک زنجیره به هم متصل نصب شده باشند. پس از فعال شدن رویداد مربوطه و ارسال پیام مورد نظر، سیستم عامل اولین حلقه از این زنجیر را فراخوانی کرده و پس از آن هر کدام از هوک‌ها باید پس از انجام کارشان، حلقه بعدی را توسط تابع CallNextHookEx فراخوانی کنند. بدیهی است که اگر این عمل به درستی انجام نشود، زنجیره مذکور قطع شده و هوک‌های بعدی هیچ پیامی دریافت نمی‌کنند. به عنوان مثال ممکن است که دو برنامه به طور همزمان اقدام به شنود بر داده‌های ارسالی توسط دستگاه ماوس کنند. در این میان می‌بایست هر دوی این برنامه‌ها، داده‌های مورد نیاز خود را به درستی دریافت نمایند.

شایان ذکر است که استفاده گسترده از هوک‌ها شدیداً سرعت پردازش سیستم عامل را کند می‌کند. زیرا با نصب زنجیره‌ای از هوک‌ها میزان پردازشی که سیستم در ازای ارسال هر پیام انجام

توضیح مختصری در مورد مهمترین فایل های کتابخانه‌ای ویندوز، توجه نمایید:

- Kernel32.dll: حاوی توابع غیر GUI مانند توابع مدیریت I/O و فایل ها، مدیریت حافظه، مدیریت اشیاء، process ها و نخ ها (Threads) می باشد. همچنین رابطی بین توابع سطح پایین موجود در NTDll.dll و سرویس های سیستم عامل است.

- GDI32.dll: حاوی تمام توابع مربوط به اعمال گرافیکی مثل ترسیم خطوط، قلم‌ها، فایل های Bitmap و کار با رنگ ها می باشد.

البته تمام توابع بصری سطح پایین سیستم در فایل Win32k.sys قرار داشته و توابع موجود در GDI در حین اجرا، آن ها را فراخوانی می کنند.

- User32.dll: تمام توابع سطح بالای مرتبط با GUI مانند توابع کار با فرم ها، پنجره‌ها، منوها و کنترل ها در این کتابخانه قرار دارند. البته این کتابخانه برای انجام اعمال بصری خود از توابع GDI استفاده می کند.

۳-۱ استفاده از رجیستری
یکی از روش های تزریق کد به برنامه‌ها، استفاده از کلید زیر در رجیستری است:

HKLM\Software\Microsoft\WindowsNT\
CurrentVersion\Windows\AppInit_DLLs

تمام DLLهایی که نام آن‌ها در این کلید ذکر شده باشد به عنوان بخشی از کتابخانه User32.dll محسوب شده و با استفاده از تابع LoadLibrary به درون حافظه تمام process ها بارگزاری می گردند. البته این روش هم مشکلاتی دارد از جمله این که فقط در مورد برنامه‌هایی می تواند به کار برود که از کتابخانه ذکر شده استفاده می کنند و به عبارتی برنامه‌های کنسولی را شامل نمی شود، زیرا این برنامه‌ها معمولاً هیچ یک از توابع User32 را Import نمی کنند.

مشکل دیگر این است که برای غیر فعال کردن این روش می بایست کلید مربوطه در رجیستری پاک شده و سیستم Restart گردد. همچنین اضافه شدن DLL مربوطه به حافظه تمام process ها، سرباری به وجود می آورد که باعث اشغال حافظه سیستم و کند شدن پردازش می شود.

۳-۲ استفاده از DLL واسطه
مثلاً فرض کنید که می خواهیم بر فراخوانی ها و پارامترهایی شنود کنیم که توسط برنامه IE به فایل کتابخانه‌ای Wsock32.dll ارسال می گردند. این DLL حاوی توابع ویندوز جهت کار با WindowsSocket و ارسال و دریافت داده‌ها در شبکه است. به عبارتی تمام برنامه‌هایی که نیاز به ارتباط با شبکه از طریق WindowsSocket داشته باشند، از توابع موجود در این فایل استفاده می کنند. حال می بایست فایلی DLL مشابه این فایل درست کرده و تمامی توابع مورد نظر را در آن قرار داده و فایل

GetMessage و یا PeekMessage اقدام به خواندن یک پیام از صف پیام های دریافتی خود می کند. قبل از پردازش پیام دریافت شده توسط برنامه، این هوک فعال می گردد.

WH_JOURNALPLAYBACK: از این هوک برای اجرای دنباله‌ای از رویدادهای ماوس و کیبورد که قبلاً توسط هوک دیگری با نام WH_JOURNALRECORD از سیستم ضبط شده‌اند، استفاده می گردد. پس از نصب این هوک، ورودی‌های استاندارد ماوس و کیبورد غیر فعال شده و دنباله مذکور اجرا می شود. همچنین نوع تابع آن JournalPlaybackProc است.

WH_JOURNALRECORD: تابع آن JournalRecordProc بوده و برای ضبط دنباله ذکر شده مورد استفاده قرار می گیرد. کاربرد این دو هوک در برنامه‌های Macro Recorder بوده که در ابتدا اعمال انجام شده توسط ماوس و کیبورد را ضبط نموده و سپس در زمان مناسب دوباره اجرا می نمایند.

WH_KEYBOARD: نوع تابع آن KeyboardProc بوده و تمام رویدادهای صفحه کلید را شنود می کند. به عبارتی هر وقت که برنامه‌ای با استفاده از GetMessage یا PeekMessage اقدام به پردازش پیامی از نوع WM_KEYUP یا WM_KEYDOWN کند، هوک مذکور آن پیام را دریافت خواهد کرد.

WH_MOUSE: نوع تابع آن MouseProc بوده و تمام رویدادهای دستگاه ماوس را شنود می کند.

WH_MSGFILTER و WH_SYSMSGFILTER: نوع تابع آن‌ها MessageProc بوده و تمام پیام‌هایی که توسط DialogBox، MessageBox، Scrollbar و منوها ارسال می گردند را دریافت می کنند.

WH_SHELL: پیام های مربوط به برنامه‌های کنسولی را دریافت کرده و نوع تابع آن ShellProc است. استفاده از تابع SetWindowsHookEx به همراه پیام‌های ذکر شده، روش استاندارد سیستم عامل ویندوز برای استفاده از هوک ها و شنود می باشد. حال آن که ممکن است بخواهیم بر روی پیام های دیگر و یا بر روی توابع موجود در فایل های DLL شنود نماییم. این مبحث در بخش بعدی تشریح خواهد شد.

۳- شنود بر توابع API

تا کنون روش هایی متفاوتی برای شنود بر توابع API توسط برنامه نویسان مختلف ارائه شده است که برخی از آن‌ها در این بخش تشریح می شوند [۳]. اما قبل از ادامه این بحث بهتر است به

برنامه و با رونویسی مکان های اجرای فراخوانی در حافظه انجام داد. قسمت مشکل این کار پیدا کردن یک به یک این نقاط در کد برنامه می باشد که کار دشواری است.

۳-۵ تغییر در IAT (Import Table Address)

این روش بر این اصل بنا نهاده شده که فایل ها اجرایی ویندوز و DLL ها همگی دارای فرمت PE هستند. در این فرمت، فایل های اجرایی از بخش های منطقی مجزایی به نام سکشن تشکیل شده اند. هر کدام از سکشن ها حاوی نوع خاصی از داده ها می باشند. مثلاً سکشن text حاوی کد اجرایی کامپایل شده برنامه بوده و یا سکشن rsrc حاوی ریسورس های مورد استفاده برنامه مثل تصاویر، اصوات یا فرم ها است.

در میان تمام آن ها، سکشن idata حاوی جدول آدرس های وارد شده (IAT) می باشد. این جدول حاوی آدرس نسبی تمام توابع Import شده به فایل اجرایی است.

پس از اجرای یک برنامه، سیستم عامل آن را درون حافظه بارگذاری کرده و این آدرس ها را به آدرس صحیح توابع مذکور در حافظه تغییر می دهد. دلیل وجود این جدول این است که فایل های اجرایی همیشه در مکان ثابتی از حافظه بارگذاری نمی شوند.

در نتیجه پس از هر بار اجرای برنامه، می بایست در تمام فراخوانی های موجود، آدرس نسبی تصحیح گشته که این عمل بسیار مشکل است. این مشکل با مجتمع کردن تمام آدرس های وارد شده در مکانی از فایل به نام IAT حل می شود.

بدین منظور کافی است که هر یک از فراخوانی ها را به طور نسبی (محلی) نسبت به IAT آدرس دهی کنیم و آدرس های اصلی را در خود IAT نگه داریم. بدین شکل سیستم عامل به سادگی می تواند یک بار آدرس مربوطه را در IAT تغییر دهد که به ازای آن تمام فراخوانی ها از طریق همین آدرس تصحیح شده انجام خواهند شد [۴].

با توجه به مطالب ذکر شده، در این روش می توانیم کد اجرایی شنودگر را در فضای process مربوطه در حافظه (Memory context) بارگذاری کرده و سپس آدرس تابع مورد نظر را در IAT، به آدرس شنودگر تغییر داده و در خود شنودگر تابع اصلی را فراخوانی کنیم.

بدین شکل قبل از این که تابع اصلی فراخوانی شود، محتوای پارامترهای ورودی آن به شنودگر ارسال می شوند که می توان تغییرات دلخواه را روی آن ها اعمال نمود. تنها مشکلی که در این زمینه پیش می آید نحوه بارگذاری کد اجرایی شنودگر در فضای process مربوطه در حافظه است.

۴- نگاهی کوتاه به ساختار فایل PE

شرکت مایکروسافت در زمان عرضه ویندوز NT، فرمت جدیدی را برای فایل های اجرایی ویندوز با نام PE (Portable Executable) معرفی نمود. کلمه PE به معنی اجرایی قابل حمل

مذکور را در فهرست حاوی برنامه مورد نظر قرار دهیم.

از این پس هر وقت برنامه مورد نظر اجرا شود، در صورت نیاز به فراخوانی توابع موجود در DLL، اولویت اول با DLL ای است که در دایرکتوری خود برنامه قرار دارد.

در صورتی که این فایل موجود نباشد مسیرهای System، System32 و غیره جستجو خواهند شد. بدین شکل تمام فراخوانی های برنامه به این DLL رسیده و می توان اقدام به شنود یا تغییرات دلخواه در پاسخ به برنامه نمود.

البته نقطه ضعف بسیار بزرگ این روش این است که می بایست تمام توابع موجود در DLL اصلی، عیناً در DLL واسطه نیز وجود داشته باشند، تا امکان استفاده از آن ها توسط برنامه وجود داشته باشد. در نتیجه ممکن است برای مانیتور کردن یک تابع مجبور شویم بیش از صدها تابع دیگر را نیز تعریف نماییم.

برای رفع این مشکل می توان با بررسی دقیق برنامه مربوطه، تمام فراخوانی های مربوط به DLL مربوطه را پیدا کرده و فقط توابع Import شده را بازسازی کنیم که خود این مسئله هم ممکن است بسیار وقت گیر و مشکل باشد.

۳-۳ تغییر در خود تابع API

برای این کار روش های مختلفی وجود دارد که یکی از آن ها دیباگ DLL مربوطه در حافظه و تنظیم بایت اولیه API با فرمان Breakpoint (وقفه Int3) می باشد. بدین شکل هر فراخوانی از تابع مربوطه، منجر به تولید یک استثنا (Exception) شده که به برنامه دیباگر ما ارسال می گردد.

مشکلات این روش، کند شدن سیستم عامل به خاطر درگیر شدن با سیستم مدیریت استثنا در ویندوز بوده و دیگری مشکلی است که زمان خاتمه process دیباگر پیش می آید. همانطور که می دانید پس از پایان process یک دیباگر، تمام برنامه های تحت کنترل آن دیباگر نیز بسته خواهند شد.

راهکار دیگر تغییر کنترل اجرای تابع با استفاده از فرمان های پرش JMP یا CALL به قسمت شنودگر است. بدین شکل که می توان تابع مربوطه را Dissassemble کرده و در ابتدای آن، فرمان پرش را قرار دهیم. مشکلی که در این زمینه ممکن است پیش بیاید کمبود فضا برای درج فرمان پرش است. زیرا ممکن است روتین API مربوطه حجم بسیار کمی مثلاً در حدود ۴ بایت داشته باشد، حال آنکه خود فرامین پرش، پس از کامپایل به ۵ بایت تبدیل می شوند.

مشکل دیگر دشواری حذف روتین شنود است. زیرا برای این کار می بایست کد برنامه در حافظه و به طور پویا رونویسی شده و فرمان پرش از بین برود.

۳-۴ تغییر فراخوانی ها

برای شنود بر فراخوانی های یک برنامه، نقطه دیگری که می توان بر آن متمرکز شد خود فراخوانی ها در فایل برنامه مذکور هستند. این کار را می توان با تغییر در فایل اجرایی برنامه و یا پس از اجرای

حافظه می‌باشد. با استفاده از این مکانیسم دیگر نیازی به تغییر آدرس در خود فراخوانی های تابع در کد برنامه نبوده و تمام آدرس‌دهی‌ها به طور نسبی و نسبت به IAT انجام می‌شوند.

۵- نتیجه

همانطور که ذکر شد، استفاده از هوک‌ها، جزئی اجتناب ناپذیر از برنامه‌های سیستمی می‌باشد. در این زمینه برنامه‌نویسان زیادی اقدام به انتشار کامپوننت‌ها و کتابخانه‌هایی از توابع، جهت تسهیل روند هوک کردن توابع سیستمی کرده‌اند.

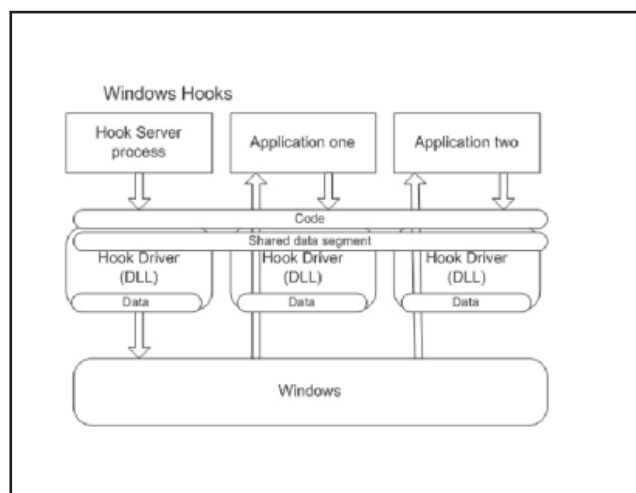
یکی از کتابخانه‌های معروف در این زمینه MadshiSDK برای پیاده‌سازی در زبان دلفی و دیگری مجموعه HookToolSDK از شرکت InfoProcess به منظور استفاده در برنامه‌های دات‌نت می‌باشد.

تمامی این کتابخانه‌ها به دفعات در ویروس‌های کامپیوتری مورد سو استفاده قرار گرفته و خسارات بسیاری به بار آورده‌اند. به عنوان مثال فرض کنید که ویروس مورد نظر با هوک کردن تابع CreateProcess کنترل اجرای برنامه‌ها را به دست گرفته و یا با هوک تابعی مثل WriteProcessMemory روند مدیریت حافظه را مختل کند.

در چنین مواردی، آنتی‌ویروس‌ها نیز با استفاده از روش‌های مشابهی اقدام به مقابله با آنها می‌نمایند.

در نهایت کاربرد هوک بسیار وسیع تر از موارد ذکر شده بوده و در اکثر برنامه‌های مورد استفاده کاربران، مانند فایروال‌ها، برنامه‌های گرافیکی، نرم‌افزارهای اداری، محیط‌های برنامه‌سازی، دیباگرها و غیره مورد استفاده قرار می‌گیرد.

از این رو کسب اطلاعات و دانشی نسبی در این زمینه، امری ضروری برای تمام برنامه‌نویسان محسوب می‌شود.



شکل (۱): بارگذاری درایور در حافظه process های دیگر

می‌باشد که دلیل آن این است که در میان تمام سیستم عامل‌های ۳۲ بیتی مایکروسافت قابل استفاده بوده و توسط آن‌ها پشتیبانی می‌شود [۵].

این فرمت بر اساس COFF (Common Object File Format) که فرمت فایل رایج و مورد استفاده در سیستم عامل یونیکس است طراحی شده بود. فرمت PE در سال ۱۹۹۳ توسط کمیته‌ای به نام TIS (Tool Interface Standard) متشکل از شرکت‌های اینتل، مایکروسافت، بورلند و آی‌بی‌ام به استاندارد و همگانی تبدیل شد.

فایل PE به بلاک‌هایی به نام سکشن تقسیم شده که حاوی داده‌هایی با خصلت‌های مختلف مثل داده/کد اجرایی/خواندنی/نوشتنی هستند.

این فایل را می‌توان مشابه یک دیسک منطقی فرض کرد که هدر (Header) فایل PE مثل بوت‌سکتور آن بوده و سکشن‌ها همان فایل‌های نوشته شده روی دیسک می‌باشند.

پس از هدر فایل، قسمتی به نام جدول سکشن‌ها (Section table) موجود است که مانند جدول FAT در دیسک می‌باشد. این جدول آرایه‌ای از هدر تمام سکشن‌های موجود در فایل است. به عبارتی به تعداد سکشن‌های موجود در فایل، در این جدول هدر وجود دارد. در هر کدام از این هدرها اطلاعاتی مثل خصلت سکشن مربوطه، آفست و سایز آن در فایل و نحوه بارگذاری آن در حافظه موجود است.

در نهایت پس از این جدول به خود سکشن‌ها می‌رسیم که حاوی تمام داده‌های موجود در فایل، با توجه به نوع سکشن می‌باشند.

پس به طور کلی یک فایل PE یک جدول بزرگ حاوی هدرها و سکشن‌ها می‌باشد. خلاصه‌ای از مهمترین قسمت‌های فرمت PE در جدول ۷ ذکر شده‌اند.

۴-۱ روند اجرای فایل‌های PE

پس از اجرای یک فایل PE، سیستم بارگذار ویندوز فضایی را در حافظه به آن process اختصاص داده و اقدام به بارگذاری فایل اجرایی در آن فضا می‌کند. بدین منظور جداول هدر را در آدرس پایه پیش‌فرض قرار داده و به همین ترتیب سکشن‌ها نیز با محاسبه آدرس نسبی شان (RVA) نسبت به آدرس پایه، در حافظه قرار می‌گیرند.

در همین حین خصلت‌های هر سکشن نیز با توجه به تعاریف موجود در هدر آن تنظیم می‌شوند. پس از این که تمام سکشن‌ها در حافظه قرار گرفتند، جدول توابع ورودی (Import table) بررسی شده و تمام فایل‌های کتابخانه‌ای مورد نیاز، در فضای رزرو شده توسط این process بارگذاری می‌شوند.

پس پایان بارگزاری، جدول خروجی (Export table) تک‌تک DLL‌های مذکور بررسی شده و آدرس توابع مورد نظر آن‌ها در IAT process ثبت می‌گردد.

در نتیجه IAT حاوی اشاره‌گر تمام توابع مورد نیاز process در

جدول (۱): مشخصات تابع SetWindowsHookEx

| نام پارامتر | نوع | توضیحات |
|---------------|-------------|--|
| idHook | Integer | نوع پیامی که هوک می‌شود را مشخص می‌نماید. مثلا WH_KEYBOARD |
| lpfn | TFNHookProc | آدرس تابع هوک در حافظه |
| hMod | Hinst | هندل مربوط به DLL ای که هوک درون آن قرار دارد. برای هوک‌های محلی از صفر استفاده می‌شود |
| dwThreadId | Cardinal | thread id مربوط به process ای که قرار است هوک شود. برای هوک‌های سراسری از صفر استفاده می‌شود |
| مقدار بازگشتی | HHOOK | در صورت موفقیت هندل تابع هوک در زنجیره و در غیر اینصورت Null را بازمی‌گرداند |

جدول (۲): مشخصات تابع UnhookWindowsHookEx

| نام پارامتر | نوع | توضیحات |
|---------------|-------|---|
| hhk | HHOOK | هندل تابع هوک در زنجیره |
| مقدار بازگشتی | BOOL | در صورت موفقیت مقداری غیر صفر و در غیر این صورت صفر را بازمی‌گرداند |

جدول (۳): مشخصات تابع CallNextHookEx

| نام پارامتر | نوع | توضیحات |
|---------------|----------|---|
| hhk | HHOOK | هندل تابع هوک در زنجیره |
| Code | Integer | طریقه برخورد تابع با دو پارامتر بعدی را مشخص می‌کند |
| wParam | Word | پارامتر ارسالی اول به تابع هوک با توجه به نوع آن |
| lParam | Longword | پارامتر ارسالی دوم به تابع هوک با توجه به نوع آن |
| مقدار بازگشتی | LRESULT | در صورت موفقیت مقداری غیر صفر است که تابع بعدی در زنجیره هوک بازگردانده است |

جدول (۴): مشخصات تابع LoadLibrary

| نام پارامتر | نوع | توضیحات |
|---------------|---------|--|
| lpFileName | LPCTSTR | هندل تابع هوک در زنجیره |
| مقدار بازگشتی | HMODULE | در صورت موفقیت هندل مازول و در غیر این صورت Null را بازمی گرداند |

جدول (۵): مشخصات تابع FreeLibrary

| نام پارامتر | نوع | توضیحات |
|---------------|---------|---|
| hModule | HMODULE | هندل مازول مورد نظر |
| مقدار بازگشتی | BOOL | در صورت موفقیت مقداری غیر صفر و در غیر این صورت صفر را بازمی گرداند |

جدول (۶): مشخصات تابع GetProcAddress

| نام پارامتر | نوع | توضیحات |
|---------------|---------|--|
| hModule | HMODULE | هندل مازول مورد نظر |
| lpProcName | BOOL | شماره گر به رشته ای حاوی نام تابع مورد نظر |
| مقدار بازگشتی | BOOL | در صورت موفقیت آدرس تابع مورد نظر و در غیر این صورت Null را بازمی گرداند |

جدول (۷): ساختار فایل PE

| | | |
|-----------------------|---------------------|--|
| MS-DOS information | IMAGE_DOS_HEADER | DOS EXE Signature DOS_ChkSum PE Pointer ... |
| | MS-DOS Stub Program | This program cannot be run in DOS mode |

| | | |
|---|--|--|
| Windows NT information IMAGE_NT_HEADERS | IMAGE_FILE_HEADER | PE signature (PE) NumberOfSections TimeDateStamp SizeOfOptionalHeader Characteristics ... |
| | IMAGE_OPTIONAL_ HEADER | MagicNumber SizeOfCode AddressOfEntryPoint ImageBase SizeOfImage SizeOfHeaders Checksum SizeOfStackReserve SizeOfHeapReserve [IMAGE_DATA_DIRECTORY [Export Table] [Import Table] [Resource Table] ... |
| Section table | [IMAGE_SECTION_HEADER[. . . IMAGE_SECTION_ [HEADER[N | Name VirtualSize VirtualAddress SizeOfRawData PointerToRawData Characteristics ... |
| Section Images | [SECTION[. . . [SECTION[N | :Binary data of sections "text." "data." "idata." "edata." "reloc." "rsrc." |

مراجع

- Microsoft® Developer Network - October 2003
- "An Introduction to Hook Procedures" by Zarko Gajic
- "API Spying Techniques for Windows" by Yariv Kaplan, May 2001
- "Peering Inside the PE: A Tour of the Win32 Portable Executable File Format" by Matt Pietrek, March 1994
- "An In-Depth Look into the Win32 PE file format" by Matt Pietrek, February 2002

Web Hosting

Servers Location : USA.

Backup Servers Location : Canada.

Platforms : Windows 2003.

General Features : Multi Server Load Balanced Network, Helm Multi Server Control Panel, Automatic Weekly Off-Side Backup, DNS Zone Editor, Shared/Dedicated SSL, Unlimited Domain Aliasing/Forwarding, **Dedicated Application Pool for More Security and Reliability**, Password Protected Folders, Unlimited FTP Access, Statistics, Front Page ext.

Programming Languages : ASP 3.0, ASP.NET v.1, ASP.NET v.2, **ASP.NET v3.5**, PHP 4.4.x CGI and ISAPI, **PHP 5**, Perl 5.6.x.

Advanced Programming Features : Installed ASP Components, Atlas (AJAX), Zend Optimizer, XML.

Emails : Dedicated Mail Server, POP3, SMTP, IMAP, LDAP Protocols, Premium Webmail/Organizer, HTML Mails, Spell Checker, RSS Reader, Advanced SPAM Filters, Anti Virus, Catch-All Accounts, Unlimited Email Aliasing/Forwarding, Auto Responder.

Databases : MS Access, MS SQL Server 2000, **Microsoft SQL Server 2005**, MySQL 4.x, **MySQL 5**.

Database Tools : Enterprise Manager, PHPMyAdmin, MySQL Front, ODBC DSN.

Support : Unlimited Technical Live, Tel Support, New 24x7 Email/Ticket Support, Online Tutorials / Documents.



| Plan | Disk Space | Monthly Bandwidth | Hosted Domain(s) | Price(Yearly) |
|------|------------|-------------------|------------------|---------------|
| P2 | 100 MB | 1 GB | 1 Website | 40.000 |
| P3 | 200 MB | 2 GB | 1 Website | 65.000 |
| P4 | 300 MB | 3 GB | 1 Website | 85.000 |
| P5 | 500 MB | 5 GB | 1 Website | 105.000 |
| P6 | 1000 MB | 10 GB | 1 Website | 145.000 |
| P7 | 2000 MB | 20 GB | 2 Website | 255.000 |
| P8 | 5000 MB | 50 GB | 3 Website | 400.000 |
| P9 | 10000 MB | 100 GB | 4 Website | 650.000 |

| | | |
|---------|----------------------------------|--------|
| Domains | com, net, org, biz, info, ws, ir | 10.000 |
|---------|----------------------------------|--------|

نماینده فروش فعال از سراسر کشور پذیرفته می شود.



تلفن: ۸۳۳۰۵

دورنگار: ۸۳۳۰۵

تهران، خیابان استاد مطهری، خیابان قابوسنامه، شماره ۱۰، واحد ۴

Visit us now
www.iranhost.com